Université de Paris 6 - Pierre et Marie Curie

Thèse de doctorat
Spécialité : Informatique

La sécurité des primitives cryptographiques basées sur des problèmes algébriques multivariables : MQ, IP, MinRank, HFE

présentée par

Nicolas T. Courtois

pour obtenir le grade de Docteur de l'Université Paris 6

Soutenue le 25 Septembre 2001 devant le jury composé de :

M.	**Sami Harari**	Directeur
MM.	**Josef Pieprzyk**	Rapporteurs
	Jean-Jacques Quisquater	
	Serge Vaudenay	
Mme.	**Pascale Charpin**	Examinateur
MM.	**Marc Girault**	Examinateurs
	Michel Minoux	
	David Naccache	
	Jacques Patarin	
	Jacques Stern	

Remerciements

D'abord ceux que tout le monde oublie : je remercie d'abord tous ceux qui se sont battus pour qu'en France l'enseignement supérieur soit accessible à tous, ainsi que M. Lionel Jospin pour avoir instauré les bourses de thèse.

Je remercie tout particulièrement mon directeur de thèse Sami Harari, qui n'a pas cessé de m'encourager ainsi que mon directeur de stage de DEA, et ensuite l'initiateur et collaborateur de choix de nombreux travaux de ma thèse, Jacques Patarin. Je remercie des responsables du projet Codes de l'INRIA : Pascale Charpin et Nicolas Sendrier de m'avoir accueilli au sein du projet.

Je remercie mes rapporteurs, dont la présence est pour moi l'ultime honneur, les chercheurs de tout premier plan international, qui ont accepté la tâche ardue de lire et référer ma thèse : Josef Pieprzyk, Jean-Jacques Quisquater et Serge Vaudenay.

Je remercie de même les autres membres de Jury, personnalités de premier plan de la cryptologie et de l'informatique hexagonales : Pascale Charpin, Marc Girault, Michel Minoux, David Naccache Jacques Patarin et Jacques Stern.

Je remercie les nombreux enseignants et chercheurs français et étrangers, cryptologues ou non, qui ont su m'éclairer par leur enseignement et leur exemple, m'épauler avec un soutien spontané en toutes choses, venir dans les séminaires, conférences et colloques, partager leur savoir, commenter mon travail.

Je tiens à remercier tout particulièrement Adi Shamir, Jacques Patarin, Louis Goubin, Nicolas Sendrier, Matthieu Finiasz, Alexander Klimov et Ernst Gabidulin pour leur qualités d'écoute, leur patience et leur soutien dans le travail sur les articles communs.

Ainsi de nombreuses personnes, dont je ne peux citer qu'une partie, tant par leurs qualités scientifiques que humaines, m'ont permis d'avancer dans la thèse. Je remercie tout particulièrement Victor Shoup, Moti Yung, Jean-Charles Faugère, Pascal Véron, Pierre Loidreau, Daniel Augot, Jacques Wolfmann, Jean-Pierre Zanotti, Louis Goubin, Adi Shamir, Gilles Brassard, Dan Boneh, Claude Carlet, Brigitte Vallée, Miklos Santha, François Morain, Jean-Marc Steyaert, Thierry Berger, Anne Canteaut, Robert Cori, David Pointcheval, Pascal Pailler, Guillaume Huysmans, Jung-Soo Byun, Patrice Vilchez et Cyril Prisette. .

Je remercie tout le personnel de l'Université de Toulon et du Var, de l'Université Paris 6, de Bull CP8 et de l'INRIA Rocquencourt pour leur soutien matériel, administratif et humain durant mon DEA et ma thèse.

Table des matières

Abstract (English)

Modern public-key cryptography depends on a handful of algebraic problems, trying to achieve the best possible security in theory, while remaining extremely efficient and practical. The original RSA scheme [1978] requires quite large block sizes (e.g. 1024 bits). Still alternatives with small size have been proposed : Elliptic Curves [1985], McEliece in Niederreiter version [1986], and recently a large family of quadratic multivariate schemes such as HFE [1996].

Multivariate cryptography provides extensive possibilities to build schemes based on a basic algebraical problem that can be supplemented with several security-improving combinatorial layers. The present PhD thesis is devoted to all aspects of Multivariate Cryptography, seen in terms of solving or applying one of the four major algebraical problems called : MQ, IP, MinRank and HFE :

1. **MQ** is the problem of solving m **M**ultivariate **Q**uadratic equations with n variables over a finite field that underlies the security of all studied schemes. Our new algorithm XL is expected to be polynomial when $m = \varepsilon n^2$, $\varepsilon > 0$ and subexponential when $m \approx n$. New algorithms for special case $m << n$ and for subfield MQ are presented.
2. **HFE** (**H**idden **F**ield **E**quation) is a problem of solving a hidden univariate equation over a finite field that is concealed with affine transformations. The reference problem is the 80-bit Patarin's HFE Challenge 1. The Shamir-Kipnis attack from Crypto'99 that reduces HFE to MinRank and MQ works in 2^{152}. We improve it and present other new attacks in 2^{62}.
3. **IP** (**I**somorphism of **P**olynomials) is the problem of finding two variable changes that relate two given sets of multivariate equations. Our new algorithms are in $q^{n/2}$ instead of $q^{\mathcal{O}(n\sqrt{n})}$ before.
4. **MinRank** is the problem of finding a linear combination of given matrices that yields a small rank. MinRank is NP-complete and generalizes well known hard syndrome decoding problems. We propose new attacks on MinRank that remain however fully exponential.

Our key result is a new, very efficient zero-knowledge authentication scheme based on the NP-complete problem MinRank.

Résumé (Français)

La cryptographie à clef publique moderne dépend d'une poignée de problèmes algébriques difficiles pour tenter d'arriver à la meilleure sécurité théorique possible, tout en restant très efficace et pratique. Le schéma RSA original [1978] demande de tailles de blocs assez grands (p.ex. 1024 bits). Des alternatives avec des tailles plus petites ont pourtant été inventées : Courbes Elliptiques [1985], la version Niederreiter de McEliece [1986], et plus récemment une large classe des schémas quadratiques multivariables tel HFE [1996].

La cryptographie multivariable dispose de vastes capacités à construire des schémas sur un problème algébrique de base, supplanté par plusieurs couches combinatoires qui améliorent la sécurité. La présente thèse est consacrée à tous les aspects de la cryptographie multivariable qui se ramène toujours à résoudre ou à appliquer un des quatre problèmes de base : MQ, IP, MinRank et HFE :

1. **MQ** consiste à résoudre m équations **Q**uadratiques **M**ultivariables avec n variables sur un corps fini. La sécurité de tous les schémas en dépend. Notre algorithme XL est polynomial en moyenne quand $m = \varepsilon n^2$, $\varepsilon > 0$ et sous-exponentiel pour $m \approx n$. D'autres algorithmes résolvent les cas particuliers de $m << n$ ou quand les coefficients sont dans un sous-corps.
2. **HFE** (**H**idden **F**ield **E**quation) consiste à résoudre une équation univariable sur un corps fini, donnée sous forme camouflée par deux transformations affines. Le problème de référence est le HFE Challenge 1 sur 80 bits. L'attaque de Shamir-Kipnis de Crypto'99 réduisant HFE à MinRank marche en 2^{152}. On explique comment l'améliorer et on présente de nouvelles attaques en 2^{62}.
3. **IP** (**I**somorphisme de **P**olynômes) est le problème de trouver deux changements de variables qui relient deux ensembles d'équations multivariables. Notre nouvel algorithme donne donne $q^{n/2}$ au lieu de $q^{\mathcal{O}(n\sqrt{n})}$.
4. **MinRank** consiste à trouver une combinaison linéaire de matrices données qui aurait un petit rang. Il est NP-complet et généralise des problèmes célèbres tel que le décodage de syndrome. Nous proposons des nouvelles attaques sur MinRank qui restent toutefois pleinement exponentielles.

On proposera un nouveau schéma très performant d'authentification sans divulgation de connaissance basé sur le problème NP-complet MinRank.

Streszczenie (Polski)

Współczesna kryptografia klucza publicznego polega na garstce problemów algebraicznych, starając się osiągnąć jak najlepsze bezpieczeǹstwo w teorii i jednocześnie pozostać nadzwyczajnie skuteczną w praktycznych zastosowaniach. Oryginalny system RSA [1978] wymaga niestety dzisiaj użycia dość dużych bloków (n.p. 1024 bitów). Na szczęście zaproponowano kilka alternatywnych rozwiązaǹ : Krzywe Eliptyczne [1985], system McEliece'a w wersji Niederreiter, i niedawno systemy drugiego stopnia wielu zmiennych z rodziny HFE [1996].

Kryptografia wielu zmiennych daje szerokie możliwości budowania kryptosystemów opartych o pewien podstawowy problem o naturze algebraicznej, ulepszone przez kilka warstw o naturze kombinatorycznej które podnoszą bezpieczeǹstwo. Niniejsza praca doktorska jest poświęcona wszystkim aspektom kryptografii wielu zmiennych, które zawsze sprowadzają się do stosowania lub rozwiązywania jednego z czterech podstawowych problemów algebraicznych : MQ, IP, MinRank i HFE :

1. **MQ** (**M**ultivariate **Q**uadratic) to problem rozwiązywania układów m równaǹ kwadratowych z n zmiennymi na ciele skoǹczonym, kapitalny dla wszystkich studiowanych systemów. Nasz nowy algorytm XL jest wielomianowy dla $m = \varepsilon n^2$, $\varepsilon > 0$ i subwykładniczy dla $m \approx n$. Podajemy nowe ataki dla przypadków $m << n$ i dla wersji z podciałem.
2. **HFE** (**H**idden **F**ield **E**quation) to problem rozwiązywania ukrytego równania jednej zmiennej na ciele skoǹczonym które jest przekształcone przez tajne transformacje afiniczne. Problemem odniesienia jest 80-cio bitowy tzw. HFE Challenge 1. Atak Shamir-Kipnis z Crypto'99 z Crypto'99 który sprowadza HFE do problemu MinRank wymaga 2^{152} obliczeǹ. Ulepszamy go i podajemy nowe ataki w 2^{62}.
3. **IP** (**I**somorphism of **P**olynomials) to problem znalezienia dwóch podstawieǹ które przekształcają jeden układ równaǹ kwadratowych wielu zmiennych na drugi. Nasz nowy atak daje $q^{n/2}$ zamiast $q^{\mathcal{O}(n\sqrt{n})}$.
4. **MinRank** to problem znalezienia liniowej kombinacji macierzy, której rząd spada. Jest on NP-zupełny, uogólnia słynne problemy kodów korekcyjnych (SD i rank-SD). Przedstawiamy nowe ataki na MinRank, które pozostają mimo wszystko w pełni wykładnicze.

Kluczowym osiągnięciem jest nowy, bardzo wydajny algorytm uwierzytelnienia Zero-knowledge oparty na problemie NP-zupełnym MinRank.

Mode d'emploi du document

Comme cele est indiqué dans le résumé, rédigé en trois langues dans les pages 15, 17 et 19, le présent travail est structuré autour de 4 problèmes algébriques MQ, IP, MinRank, HFE. L'impact et les contribution de ce travail vont bien au delà de ces 4 problèmes.

Le présent manuscrit contient des parties en français, et en anglais. De manière générale, tout ce qui important, intéressant et général est en français. Certaines parties qui sont ajoutées pour complétude, et qui n'ont d'intérêt que par leur résultat, sont toutefois rédigées en anglais. 8% seulement de l'ensemble du document est en anglais.

0.1 Structure du document

Le document comporte 8 parties :

La première partie traite de la cryptologie en général qui est définie en tant que la science du secret. On explique l'évolution de la notion du secret, et l'évolution de la cryptologie elle-même au cours des siècles.

La deuxième partie traite de la cryptographie dite à clef publique. On en donnera une synthèses, avec un peu d'histoire, ses buts, ses méthodes, ces dilemmes. On expliquera la plupart des notions de sécurité existantes en cryptographie à clef publique. On présentera des principales branches de la cryptographie à clef publique et on discutera de leur avantages et les inconvénients. Cela nous motivera pour étudier une de ces branches, appelée **Cryptographie Multivariable**. Enfin, nous allons esquisser des applications industrielles de la présente thèse.

Dans les parties qui suivent, quatre sont consacrées à quatre problèmes algébriques qui fondent la cryptographie à clef publique multivariable.

Ces problèmes sont IP, MQ, MinRank et HFE.

La troisième partie est consacrée au problème MQ (Multivariate Quadratic) de résoudre les équations quadratiques multivariables. Nous présenterons notamment notre récent algorithme XL , [42] pour résoudre MQ. Nous avons effectué de nombreuses simulations sur XL. Il y a aussi de nouveaux algorithmes pour les cas particuliers de MQ avec $m << n$, et aussi pour MQ avec les coeffi-

cients dans un sous-corps de K. Ces attaques permettent de casser de nombreux schémas qu'on avait crus solides auparavant.

La quatrième partie est consacrée au problème HFE (Hidden Field Equation), et au cryptosystème du même nom [65]. Elle décrit tous les attaques connus pour cryptanalyser le cryptosystème HFE, à l'exception des attaques génériques sur MQ décrits dans 6. Notamment on décrit en détails l' attaque expérimentale sur HFE publiée aussi dans sa version de base dans [68]. C'est la meilleure attaque connue pour le problème HFE. Les résultats sont résumés dans 16.

La cinquième partie est consacrée au problème IP (Isomorphisme de Polynômes). Nous présenterons notamment notre algorithme en $q^{n/2}$, [88], certainement le meilleur algorithme connu actuellement.

La sixième partie est consacrée au problème MinRank. On définit le problème, et on explique ses relations avec d'autres problèmes importants. Dans 23.1.2 on donnera plusieurs preuves que MinRank est NP-dur. Dans 24 on étudie en détails 6 attaques sur MinRank. Si les paramètres du problème sont bien choisis, il s'avère que tous les attaques connues pour MinRank sont exponentiels. Cela nous permet de proposer dans 25.2 un nouvel algorithme d'authentification à divulgation nulle de connaissance (Zéro-knowledge) basé sur MinRank. Nous allons étudier des la sécurité et les amélioration de ce schéma, pour arriver dans 26.1.2 à la conclusion que c'est un de plus performants schémas connus basé sur un problème NP-dur.

La septième partie et la dernière traite des attaques sur les cryptosystèmes multivariables en supposant que le texte clair soit dans une langue naturelle, par exemple en français.

0.2 Des compléments de la thèse

La version la plus récente du présent document se trouve à : `http://www.minrank.org/phd.pdf`

Pour regarder le document sous forme électronique il est conseillé de télécharger la version pdf car elle contient des liens hypertexte qui facilitent la lecture.

Un certain nombre de pages web ont été rédigées par l'auteur de la présente thèse et qui sont considérées comme des compléments de la thèse. On y trouve des documents annexes, des articles cités à télécharger, des résultats supplémentaires récents ou anciens, de la vulgarisation scientifique, de la promotion des technologies qui appliquent la cryptographie à clef publique, les dernières nouvelles de la cryptographie multivariable et des liens vers d'autres sites. Les documents sont en différents langues, le plus souvent en anglais. Les adresses donnés peuvent être considérées comme permanentes.

`http://www.minrank.org/~courtois/myresearch.html`

`http://hfe.minrank.org`

`http://www.minrank.org/minrank/`

`http://www.minrank.org/mceliece/`

`http://www.minrank.org/quartz/`

`http://www.minrank.org/flash/`

`http://www.minrank.org/ttm/`

`http://www.kryptografia.org`

`http://www.kryptografia.maxi.pl`

Première partie

Sur la cryptographie

Chapitre 1

La science du Secret

La science du Secret. Rien ne définit mieux la cryptologie moderne que cette citation de Jacques Stern, titre de son livre destiné au grand public et consacré à la cryptologie [6].

Jadis, la cryptographie était définie comme l'art et la manière de créer des cryptogrammes, les messages secrets compris uniquement par leur destinataire légitime, au moyen de codes et répertoires. On disait aussi que la Cryptologie était le science des codes secrets. Mais de préférence on ne disait rien, car tout était secret...

La cryptographie était fermée et servait des objectifs fermés, dans la politique, la diplomatie et (surtout) la guerre.

Rien de tel dans la cryptologie moderne, qui est devenue progressivement de plus en plus ouverte (signe de notre temps), et qui poursuit des buts de plus en plus ouverts : la confiance, la sécurité, amélioration des échanges, transparence et vérifications faciles, équilibre de pouvoirs.

Deux révolutions scientifiques majeures ont accompagné cette ouverture : l'invention de la cryptographie dite à clef publique [Ellis, vers 1970] et l'invention de l'authentification sans divulgation de connaissance, appelée aussi Zéro-knowledge, [Goldwasser, Micali, Rackoff 1985].

Avant de parler de ces deux révolutions, on va essayer d'esquisser l'histoire de la cryptographie sous un angle particulier. On essayera de voir comment évoluèrent au fil du temps les notions du secret et la notion de la sécurité. Ces deux notions permettent d'appréhender l'ensemble de l'évolution de la cryptologie.

1.1 Notion de secret

Dans la cryptographie disons classique (l'art de faire du chiffrement) le secret est une **clef secrète** et ce nom a une signification profonde, avec des propriétés d'un véritable objet matériel. Une clef est un objet que l'on peut dupliquer, transporter, utiliser pour ouvrir un coffre, ou encore l'enfermer dans un autre coffre.

Par la suite cette métaphore a pu être dépassée, et de même que la vie devient de plus en plus immatérielle, la cryptographie fait de même. En effet,

comment continuer à appeler **clef** une quantité qui :

- peut être partagée en parts qui n'en contiennent pas le moindre morceau (partage de secret parfait)
- peut être crée à deux endroits distants sans être transportée (établissement de clef à distance Diffie-Hellman ou téléportation quantique)
- peut être utilisée en sécurité pour chiffrer sans être compromis pour déchiffrer (chiffrement avec clef publique)
- enfin, on peut convaincre n'importe qui de son existence (protocoles à divulgation nulle de connaissance, Zéro-knowledge), sans qu'il en garde la moindre trace ni preuve. C'est comme si l'on pouvait voir un objet sous tous les angles, le toucher, sans pouvoir le photographier ni le filmer pour convaincre d'autres personnes.

Aujourd'hui on appellera une **quantité secrète** une **clef** uniquement si on s'en sert effectivement pour ouvrir un quelconque 'coffre', par exemple un message chiffré. Sinon on parlera plutôt d'un **secret** ou **quantité secrète**.

On note toutefois que jusqu'à très récemment cette vision matérielle du secret était suffisante.

1.2 Étapes d'évolution de la cryptologie

1.2.1 Secret, car pas connu

Aux débuts de la civilisation était secret tout ce qui n'était pas évident, écrit noir sur blanc, et la cryptographie artisanale protégeait tant que la clef, la méthode et souvent l'idée même de faire du chiffrement restait secrète. Même l'écriture était secrète pour ceux qui ne savaient pas lire, par exemple dans l'Égypte ancien l'écriture était le privilège d'un caste très fermé.

Il y avait essentiellement deux états possibles : avoir ou ne pas avoir ce secret, comme pour un objet matériel indivisible.

La première notion de secret était donc basée sur sa (relative) inaccessibilité, guère plus, et c'est sans doute la raison pour laquelle la cryptographie, pas sûre d'elle-même, est longtemps restée secrète.

Cette notion de sécurité est naïve et sans espoir. Mais il ne faut pas croire qu'elle ne fonctionne pas.

Au contraire, on la voit tous les jours, sous les noms de la sécurité commerciale ou sécurité médiatique : basées sur l'incapacité de l'Adversaire à comprendre qu'il est capable de percer le secret s'il le voulait vraiment. Et cela marche encore assez souvent.

1.2.2 Secret et information

Par la suite les chiffres se sont complexifiés, il existait de longs répertoires chiffrants, et il fallait que les principales puissances installent des 'cabinets noirs' qui devaient intercepter beaucoup de courrier pour les reconstituer.

Les méthodes utilisées pour casser étaient artisanales, et général on y parvenait à coup sûr, dès que c'était possible : la longueur du secret, sa quantité

d'information étaient le seul obstacle.

Cette vision du secret a culminé dans le travaux sur l'information de Claude Shannon dans les années 1940, où nous avons appris à mesurer l'information. Ainsi, le secret se mesurait en nombre de bits.

Ensuite, Shannon a formalisé la redondance (qui est locale) des langues naturelles, et a calculé combien il fallait de texte(s) chiffré(s) pour être en mesure de récupérer le secret (la notion de la distance d'unicité).

Ces estimations théoriques se sont avérées plutôt exactes dans la pratique, et avec de l'intelligence mais surtout, du travail et de la diligence, on cassait tout ce que la théorie de Shannon permettait.

La deuxième notion de secret était donc sa longueur, et plus précisément l'information formalisée par la notion de l'entropie.

Cette notion a mené à la sécurité inconditionnelle. Malheureusement cette notion théorique ne s'applique pas très bien en pratique, car il faut une clef secrète très longue, et rares sont les applications où on peut se le permettre.

1.2.3 La sécurité calculatoire

Ensuite les chiffres se sont complexifiés. Toujours les mêmes, on peut dire qu'il y avait essentiellement deux chiffres antiques que l'on peut appeler 'confusion' et 'diffusion'. Sauf que l'on a compris l'importance du surchiffrement.

Car c'est en composant ces éléments simples avec de plus en plus de couches qu'on peut obtenir des chiffres qui ne soient pas cassés à coup sûr. Mais alors, et c'est arrivé vers le début du siècle, il est devenu nécessaire d'utiliser des machines chiffrantes de plus en plus complexes, par exemple Enigma.

L'intelligence humaine a toujours et encore triomphé, Enigma a été cassée, et re-cassée (on a toujours pu le faire), malgré les changements et améliorations très fréquentes. Historiquement parlant, le mérite revient sans partage aux mathématiciens et cryptologues du chiffre polonais (Rejewski, Różycki, Zygalski) qui ont **cassé** (percé le secret de) toutes les versions d'Enigma depuis fin 1932, y compris celle (fait peu connu) qui était rentrée en utilisation juste au début de la Deuxième Guerre Mondiale. Quand, en 1939, dans une conférence secrète tenue à Varsovie, les polonais ont présenté aux Anglais stupéfaits, et aux Français agréablement surpris, tous leur travaux et résultats.

Depuis ce temps, la cryptanalyse d'Enigma est devenue une véritable industrie. Les Français et les Polonais travaillaient d'un côté de la Manche, les Anglais de l'autre. Il fallait sans cesse tout refaire car la clef d'Enigma était modifiée chaque jour, et de plus on changeait aussi d'autres paramètres.

Si la collaboration a continué, les Anglais ont fini par mener la course, conséquence de moyens colossaux qu'ils y ont consacré. Près de 12 000 personnes ont travaillé à Bletchley Park vers la fin de la guerre.

L'Enigma a apporté une dimension nouvelle à la notion du secret.

En effet, peu à peu l'Enigma avait été débarrassée de ses faiblesses majeures, sur le plan des propriétés statistiquement significatives qui ont permis des premières cryptanalyses.

Elle est devenue progressivement un vrai chiffre moderne, à comportement de plus en plus proche d'une source aléatoire. Casser l'Enigma est donc de-

venu quasiment impossible. Les cryptanalyses ont été, certes, de plus en plus ingénieuses, mais l'intelligence du cryptanalyste ne suffisait plus. Il fallait de plus en plus de temps pour récupérer la clef, et les Polonais ont fini par construire une machine pour automatiser ce travail. Elle fut appelée 'bombe', à cause du bruit d'horlogerie qu'elle produisait (c'était au tout début une machine électromagnétique). Quand Alain Turing a travaillé sur les "bombes", quelques années plus tard, il a voulu en faire une machine universelle, appelée aujourd'hui une machine de Turing.

Nb. La machine de Turing était **universelle** du point de vue de puissance logique, c'est à dire qu'elle étaient capable de résoudre n'importe quel autre problème résolu par une machine.

Ainsi la notion de secret a gagné en profondeur, avec une dimension calculatoire. Par exemple, une chose qui n'est pas secrète du point de vue de l'information peut toujours rester réellement secrète.

On peut dire qu'en plus de l'entropie on a ajouté une distinction entre les secrets 'durs' et 'mous' : ceux qui ont été percés et d'autres pas par rapport à la puissance de calcul existante qui est fixée.

Plus tard, vers les années 1970, quand on a acquis une réelle confiance dans la sécurité calculatoire, il n'y avait plus besoin de maintenir la cryptologie secrète. On a commencé à publier les fonctions de chiffrement, et les gouvernements américain et russe ont proposé des normes : DES et GOST.

Trois décennies plus tard on peut dire que la confiance à ces solutions n'a pas été déçue.

1.2.4 Les modalités d'un secret

Dans la cryptologie moderne les **modalités** d'existence d'un secret sont devenues de plus en plus subtiles.

En effet il n'importe pas seulement combien d'information (au sens de la théorie de Shannon) peut obtenir un Adversaire à partir des données du problème, et de combien de puissance de calcul on dispose.

Ce qui importe surtout, c'est la façon dont on peut accéder à l'information, surtout le fait que l'on soit actif ou pas dans le processus et à quel niveau on peut interagir.

On n'est plus dans un modèle cartésien à deux dimensions, une calculatoire, l'autre de l'information, mais dans un modèle de treillis qui représente, par exemple pour une fonction cryptographique f, différents modes d'accès avec plus ou moins directs avec leur hiérarchie. On peut appeler cela les niveaux de connaissance d'une fonction.

Il va de soi qu'il y a une infinité de modes d'accès possibles, et le diagramme de la page suivante en montre quelques-uns.

Ce qu'il est important de noter, c'est qu'il n'y a plus un ordre total entre les différents niveaux, certains sont incomparables. C'est ainsi que la cryptographie est devenue un art, et non plus une question de puissance brute.

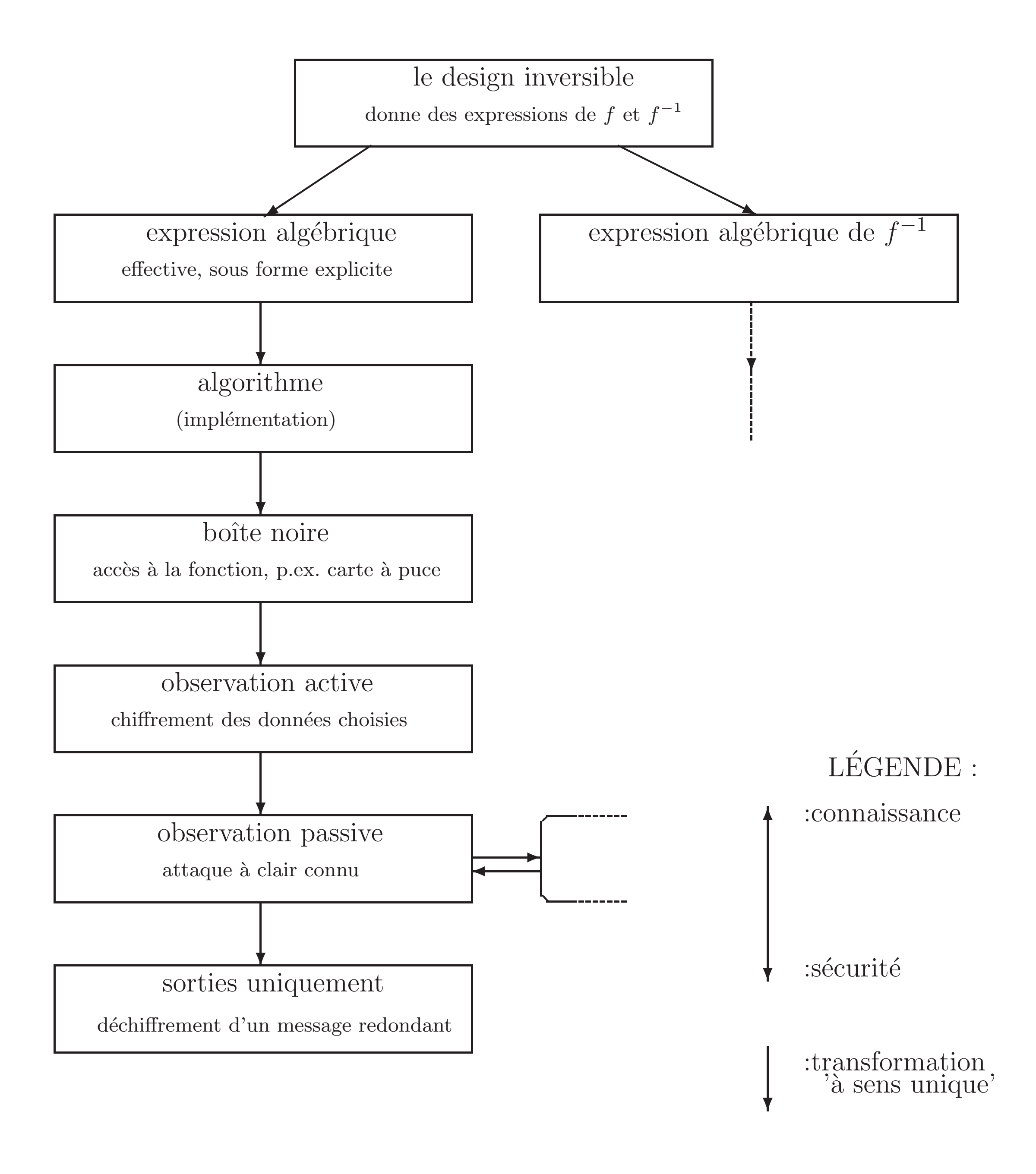


Fig. 1.1 – Les niveaux de connaissance d'une fonction de chiffrement

1.2.5 Recherche de bases solides

Avec l'invention de la cryptographie à clef publique, la compréhension du secret avait été bouleversée. En effet elle permet de communiquer sans secret préalable.

Quel type de sécurité peut-on ainsi obtenir ? Ni la science ni la pratique n'ont apporté aujourd'hui de réponse définitive. L'existence même de la cryptographie à clef publique demeure toujours une question ouverte.

Il apparaît que c'est d'abord une sécurité calculatoire, connue déjà depuis Enigma et arrivée à la maturité avec DES. La différence est que si avec DES on doit obtenir de la part de l'adversaire par exemple des couples (clair, chiffré) pour avoir une chance de cryptanalyser, dans la cryptographie à clef publique on en a strictement aucune utilité, puisque on peut les obtenir soi même en utilisant la clef publique. Ainsi on a amené la sécurité du problème du déchiffrement du message, à un problème mathématique d'inverser une fonction.

Est-ce un progrès ? Quel type de sécurité peut-on obtenir (en chiffrement) avec une fonction **publique** mais difficile à inverser ?

Peut être pas beaucoup, car même si la fonction admet des instances difficiles, elle peut être faible, voire même être faible la plupart du temps. Pour cette raison on a privilégié des systèmes où la sécurité reposait de façon prouvée sur une donnée concrète à trouver, par exemple la décomposition en facteurs premiers d'un nombre, plutôt que sur la difficulté nébuleuse d'inverser une fonction. Paradoxalement on est revenu à la notion d'un secret matérialisé !

1.2.6 Se prémunir contre toute attaque

Dans la cryptographie à clef publique on peut publier les paires (clair,chiffré) produites avec des clairs choisis sans aucun préjudice pour la sécurité du schéma. Le problème du déchiffrement du message se réduit rigoureusement à un problème mathématique.

Or ce n'est pas la seule menace contre un cryptosystème à clef publique. Les attaques les plus redoutables sont les attaques actives, par exemple celle qui demande à déchiffrer les messages choisis. On rencontre beaucoup d'attaques de ce type dans la cryptographie à clef publique.

On souhaite qu'une classe d'attaques beaucoup plus large soit réduite (également) à un problème mathématique.

La sécurité contre les attaques actives admet tous les **Adversaires** actifs, qui sont libres d'interagir avec le protocole, et qui cherchent à obtenir (même un petit) **Avantage**.

Les premiers exemples connus des cryptosystèmes solides contre les attaques actives étaient des algorithmes d'authentification sans apport de connaissance (Zéro-knowledge), apparus vers 1985.

Qu'est devenu la notion du secret ?

On a enfin enlevé au secret son aspect modal en intégrant dans le modèle toutes sortes d'attaques possibles. On est arrivé à prouver que (modulo tel et telle hypothèse) le système est solide contre **tous** les types d'adversaires.

Ainsi on est arrivé à une notion de sécurité prouvable, mais qui a fait reposer

la sécurité encore davantage sur un seul secret interne (p.ex. la factorisation). Le secret reste (en quelque sorte) matériel, un objet (mathématique) très concret.

1.2.7 Le secret de plus en plus matérialisé

Aujourd'hui personne n'arrive de façon convaincante, à baser la sécurité, sur autre chose qu'un secret matérialisé.

Qui plus est, la carte à puce et la biométrie, le matérialisent encore, et de façon irréversible semble-t-il. Plus personne ne peut aujourd'hui mémoriser un mot de passe suffisamment long, pour qu'il ne puisse être cassé par un ordinateur.

Décidément la cryptographie n'a pas su, ou n'a pas voulu, se débarrasser d'une notion de secret matérialisé. Mais...

1.2.8 La secret en cryptographie quantique

Dans la cryptographie quantique, le secret, bien que matériel, a un comportement qui échappe à une vision matérialiste du monde. On peut par exemple créer des quantités secrètes à des endroits éloignés, comme avec le protocole connu de Diffie-Hellman, mais sans aucune hypothèse sur la puissance de calcul de l'Adversaire.

La sécurité ainsi obtenue reste controversée.

Admettons d'abord l'interprétation dite de Copenhague de la mécanique quantique : il n'y a rien d'autre que les phénomènes mesurables et les lois abstraites de la mécanique quantique fonctionnent, qu'elles soient contraires au bon sens ou non. On peut alors garantir la sécurité des protocoles quantiques contre un attaquant classique, qui fait de mesure locales dans l'espace et dans le temps, et avoir une sécurité encore plus forte que donne Zéro-knowledge.

Malheureusement, on peut imaginer un adversaire quantique : un grand ordinateur quantique qui réagirait avec l'ensemble du protocole (sans faire de mesures locales), qui serait décrit par une fonction d'onde qui ne s'effondre pas au cour du protocole, ni même par la suite ; par exemple la fonction ne donnerait même pas le secret, mais casserait directement un chiffre qui utiliserait ce secret.

On pense donc que si la sécurité des protocoles quantiques est prouvée contre les adversaires classiques, elle n'est pour autant garantie à jamais.

1.3 Les nouveaux visages de la cryptologie

On a vu que la cryptologie a quitté le monde simpliste soumis à des lois de la mécanique, la matière et la force, et est arrivé dans un monde où règnent avant tout l'intelligence, la complexité et l'interaction. Un monde étonnant, qui bouleverse notre vision du monde car il permet d'en dépasser les principales contradictions.

Ainsi on peut définir la cryptologie moderne comme :

La cryptologie est la science qui cherche à connaître et étudier les lois et les principes qui gouvernent le secret, la confiance et la sécurité dans les interactions.

Enfin, il convient de redéfinir **la cryptographie** comme l'art d'agir en présence des lois du secret et d'atteindre de façon efficace les buts classiques : **l'intégrité, la confidentialité, et l'authentification**. On pourrait résumer l'ensemble sous un terme plus large du calcul réparti sécurisé.

Quant au secret, il est un outil pour garantir la sécurité, et la sécurité sert à construire la confiance (même avec des adversaires supposés malhonnêtes).

Bref, la cryptographie est un art de faire du calcul réparti et de maximiser la confiance sous les contraintes matérielles et humaines réalistes.

Avec ses Adversaires, ses attaques, défenses et protections, c'est aussi un **art martial**, mais pas un art de combat ni de guerre (électronique).

En effet, sa finalité n'est pas de combattre sans fin, ni de détruire l'Adversaire mais de chercher à se prémunir totalement contre toute sorte d'attaque.

1.3.1 Cryptographie, politique, modernité

La cryptographie permet ainsi d'atteindre une plus grande ouverture, basée sur la confiance, et un meilleur équilibre des pouvoirs. La cryptographie est donc un des piliers de la démocratie.

Lionel Jospin, en libéralisant la cryptologie à son arrivée au gouvernement, a sorti la France d'une politique de sous-développement programmé (restriction à 40 bits) dans la quelle elle avait été plongée et ainsi a rendu un service important à la nation.

La cryptologie sert à réguler et civiliser l'ère du partage et d'échanges d'information dans laquelle nous entrons. Si l'internet permet à l'information de circuler sans entrave, la cryptologie la régule, en fait une véritable civilisation.

Deuxième partie

La cryptographie à clef publique et applications

Chapitre 2

La cryptographie à clef publique

2.1 Définitions

Dans la cryptographie classique, appelée aujourd'hui **cryptographie symétrique** ou **cryptographie à clef secrète**, la sécurité des données ne peut être atteinte qu'à condition qu'il existe un canal physiquement sûr, qui permet un échange préalable de clef. Par exemple : une valise diplomatique, rencontre sans témoins dans un parc public, échange de disquette avec clef de main en main.

La **cryptographie à clef publique** se définit comme permettant de communiquer en sécurité **sans** une clef préalablement échangée, et toujours en présence (d'écoute par) de tierces personnes. On pourrait même admettre que le canal de communication est totalement public. Par contre cela n'est possible qu'à condition d'existence préalable d'un canal public **public non-modifiable**, que personne ne peut empêcher ou modifier. Sur ce canal on peut envoyer une quantité appelée **clef publique** que l'on peut par la suite utiliser pour chiffrer et authentifier les échanger de façon sûre, en utilisant des utils de la cryptographie à clef publique.

Un grand avantage de ce type de schémas est que la **clef secrète** correspondante à la **clef publique** d'une personne n'est pas partagée, elle ne quitte jamais son propriétaire. Il est donc plus facile de s'assurer qu'elle n'est pas compromise. Pour cette raison on distingue parfois la **clef privée**, une notion plus forte que celle d'une clef secrète. A l'opposé, la **clef publique** est largement distribuée. C'est cette asymétrie dans la distribution de clefs qui fait que la **cryptographie à clef publique** est souvent appelée **cryptographie asymétrique**. Les deux parties n'ont pas les mêmes quantités secrètes.

2.1.1 Histoire de la cryptographie asymétrique

Version officielle - le concept aurait été inventé et publié en 1976, par Diffie et Hellman, et indépendamment par Merkle. Le premier cryptosystème praticable est le RSA publié en 1977 par Rivest Shamir et Adleman.

Mais D'après le chiffre britannique : Communications-Electronics Security Group (CESG) la cryptographie à clef publique avait été inventée par un des piliers du CESG, James Ellis. En Janvier 1970 il publie un premier article (confidentiel) dans lequel il établit que c'est possible. Par la suite en 1973 Clifford Cocks invente une variante du futur RSA, et en 1974 Malcolm Williamson trouve un moyen d'échange de clefs, variante du futur Diffie et Hellman. Les détails sont décrits par Ellis dans un article de 1977 publié après sa mort en 1997 [2]. C'est Clifford Cocks lui-même, qui pendant une session-surprise durant la 6-ème conférence IMA à Cirencester, le 17 Décembre 1997, qui dévoilera la vérité, après près de 30 ans de silence. Les détails sont publiés sur le site web de CESG [2].

2.1.2 Histoire de la cryptographie multivariable

Beaucoup de cryptosystèmes de nature **combinatoire** sont apparus dès la début de la recherche publique en cryptographie. De tels cryptosystèmes basés sur les sacs-à-dos, le graphes, etc.. ont fleuri et souvent ont été compromis par des attaques tels que la démonstration spectaculaire de la cryptanalyse des sacs-à-dos par Adi Shamir à la conférence Crypto'82.

Mais en même temps il apparaît de schéma à caractère plus **algébrique**, qui utilisent des grands ensembles d'équations **multivariables linéaires** sur de corps finis, souvent regardées en termes de codes correcteurs. En commençant par McEliece qui est un de plus vieux cryptosystèmes à clef publique, proposé en 1978, [127] et la variante de Niederreiter [128], beaucoup d'autres schémas existent dont on citera seulement les meilleurs : [121, 131, 133, 152, 137, 153, 139] ainsi que le récent schéma de signature à la base de McEliece [116] proposé par Finiasz, Courtois et Sendrier. Cette branche a résisté a des décennies de cryptanalyse et contient de nombreux schémas pour lesquels la meilleure attaque connue est exponentielle. Deux problèmes semblent particulièrement durs : le problème SD qui est à la base de la sécurité de nombreux cryptosystèmes [127, 137, 139, 135, 134, 116]. Un autre problème difficile est le problème MinRank, défini dans 22.1 dont dépend la sécurité des nombreux autres schémas tels que [121, 131, 133] ainsi que de HFE et du cryptosystème MinRank décrit dans la présente thèse.

Ensuite il y a toute une série de schémas **multivariables quadratiques** sur de corps finis. Ce courant semble apparaître vers 1983 au Japon avec Matsumoto et Imai [52, 53, 54, 51] puis aux USA avec Fell-Diffie [48] et Cade. Par la suite de nouveaux cryptosystèmes de ce type seront proposés par Shamir en Israël [75] et enfin en France par Jacques Patarin [65, 59], avec Louis Goubin [60, 72] et Aviad Kipnis (Israël) [76], ainsi que par moi-même [133, 69] et par Patarin, Goubin et moi-même [61, 66, 62, 67].

Peu à peu, certains schémas, notamment des variantes de HFE, montrent leur exceptionnelle résistance aux attaques due à l'existence à la fois d'une couche **algébrique**, reliée également au très difficile problème MinRank, doublée d'une ou plusieurs couches **combinatoires** (voir 3.2.3) qui introduisent des perturbations mettant à mal le peu d'attaques qui existent contre le sous ensemble algébrique de base.

2.2 Les tâches de la cryptographie à clef publique

La cryptographie à clef publique ne se limite pas au chiffrement et répond à d'autres besoins importants qui ont en commun l'absence de quantité secrète commune aux 2 (ou plus) interlocuteurs du schéma.

[1] **Établissement de clef secrète** : Deux personnes discutent via un canal publique que personne ne peut empêcher ou modifier. Elles veulent établir une quantité secrète commune (**une clef de session**) inconnue des tierces personnes qui écoutent sur la ligne. La solution à ce problème avait été inventée en 1976 par Diffie et Hellman. Par contre si on est pas sûr de l'integrité du canal, et si quelqu'un peut s'introduire entre les deux correspondants, ce n'est plus possible (man-in-the-middle attack). On a besoin d'avoir une quantité secrète préalable pour authentifier [3] toutes les communications.

[2] **Chiffrement à clef publique** : Comme déjà décrit, il faut que toute personne puisse chiffrer un message avec une **clef publique**, et une seule personne puisse le lire avec la **clef secrète** correspondante. On peut obtenir [2] avec [1] et vice versa.

[3] **Identification, Authentification des personnes** : Prouver que c'est bien la personne autorisée qui veut accéder à un immeuble, à un système distant etc.. (décision en temps réel).

[4] **Authentification des messages, Certification** : Permet de garantir qu'une personne donnée est l'auteur d'un message ou document (certifie la provenance). Elle diffère de [3] car dans l'identification il n'y a pas de message.

[5] **Signature numérique** : Garantit la provenance d'un message, comme dans [4], mais en plus la signature doit pouvoir constituer une preuve **indéniable** (p.ex. devant la justice), de son authenticité. De plus les paramètres doivent être suffisamment sûrs, pour que la signature garde sa valeur pendant des dizaines d'années.

Qui peut le plus, peut le moins ; la signature numérique peut également assurer [3] et [4]. De même la cryptographie classique (symétrique) peut, par exemple avec le DES, assurer le [3] et le [4].

Par contre la possibilité de faire des signatures numériques [5] était inconcevable avant l'invention de la cryptographie à clef publique.

[6] **Authentification à divulgation nulle de connaissance** : doit permettre [5] mais sans avoir la moindre chance de convaincre une tierce personne de ce dont on se convaincra en interagissant avec un Prouveur légitime (Zéro-knowledge).

[6] permet aussi d'assurer [5] en utilisant des fonctions de hachage quoi remplace les questions du Vérifieur (queries). Cela s'appelle l'heuristique de Fiat-Shamir [157].

[7] **Problèmes répartis :** Par exemple le vote électronique...

2.3 La sécurité

Le sécurité, être sûr, cela ne veut rien dire en soi. Est-ce qu'un ordinateur peut être sûr contre une grenade qui explose au dessus du CPU [Schneier] ? La sécurité ne peut-être que relative. De bonnes questions à poser sont : Sûr par rapport à qui ? Sûr par rapport à quoi ? La sécurité de tout cryptosystème dépend en toute généralité de 3 données :

1. Quels sont **les ressources** de l'Adversaire ? Cela concerne sa puissance de calcul, sa mémoire, et autres paramètres important concernant la technologie accessible, par exemple Adversaire classique ou quantique. Dans la vie réelle il faut aussi poser la question du risque que l'adversaire est prêt à assumer.
2. Quelle **notion de sécurité** on cherche à obtenir : Quel est **le but de l'attaquant** ?

   En partant de buts simples et évidents, tels par exemple déchiffrer tout message chiffré avec le cryptosystème donné, la science cryptographique a cherché de définir des attaques plus générales, qui englobent toutes les attaques simples que l'on peut imaginer. Une des plus fortes notions de ce type est la notion d'indistinguabilité souvent notée IND dans des différents contextes. Le but de l'attaquant est de distinguer le cryptosystèmes, ou les données obtenues au cours de son utilisation, d'un objet idéal qui est réellement sûr (par définition), ou qui tout au moins est réputé sûr. Cette notion est en effet très générale, car si l' on sait déchiffrer, on sait par la même distinguer d'un objet sûr. C'est également très puissant pour faire des preuves de sécurité.
3. Quel est **scénario d'attaque**, c'est à dire quel **type d'accès ou d'interaction** peut-il avoir avec le cryptosystème dont il cherche à percer le secret.

   Ceci décrit l'accès que peut avoir l'adversaire au secret contenu dans la fonction, voir la figure 1.1. Le secret peut d'ailleurs être contenu seulement dans sa conception. Un scénario d'attaque peut aller d'une simple possibilité d'observation des messages chiffrés sans en connaître le clair, jusqu'aux scénarios très puissants ou l'adversaire va disposer d'un accès total à la fonction sous forme d'une boîte noire (un oracle) ou encore plus fort, à une formule mathématique lui permettant de calculer les expériences avec l'oracle lui même. Certaines de ces possibilités sont décrites dans la figure 1.1, page 31.

2.3.1 Preuves de sécurité ?

Aucun schéma à clef publique n'a malheureusement été prouvé totalement sûr. La sécurité prouvable consiste à prouver la sécurité relativement à un problème calculatoire réputé difficile, tel que la factorisation ou le décodage d'un code correcteur aléatoire. Toute preuve de sécurité sera alors faite pour rapport à un triplet de conditions préalables bien précises :

1. Les ressources de l'adversaire. Cette condition sera en réévaluation permanente avec le progrès de la technologie informatique.
2. Selon le but que l'adversaire cherche à atteindre, on aura tel ou autre **notion de sécurité** par exemple la sécurité sémantique (IND).
3. Sous condition que l'attaque aura lieu dans un cadre bien précis. Dévier du **scénario d'attaque** et avoir un **accès** même très indirect autre que prévu à la fonction peut tout changer.

Il va de soi que l'on cherchera à prouver la sécurité contre les adversaires aussi puissants que possibles (1), par rapport à la **notion de sécurité** (2) la plus forte possible et dans un **scénario d'attaque** le plus général possible (3). Cela n'est pas toujours possible, et la sécurité peut demander des compromis.

2.3.2 Preuves et arguments

Certains auteurs rechignent à appeler **preuve** tout ce qui, à part une réduction vers un problème connu, dont la difficulté constitue l'axiome principal utilisé, demande des hypothèses en plus sur d'autres objets. On appelle **arguments** les preuves qui contiennent de telles hypothèses supplémentaires.

Par exemple dans un schéma incorporant une fonction à sens unique, on supposera que la sortie de la fonction se comporte comme un oracle aléatoire. Cette hypothèse, proposée par Bellare et Rogaway, et appelée *Random Oracle methodology, RO* est critiquée [14]. En même temps elle continue d'avoir énormément de succès car elle permet de prouver la sécurité de nombreux schémas. Des nombreux chercheurs restent convaincus que les preuves faites dans ce cadre assurent que les schémas en question sont corrects (exempts de défauts) et sûrs en pratique [21, 17].

Par exemple dans [116] nous prouvons prouve la sécurité dans le modèle de l'oracle aléatoire RO de la sécurité de 4 schémas de signature à base de McEliece que nous avons proposé avec Matthieu Finiasz et Nicolas Sendrier. Dans cette preuve [116], on va substituer les réponses que va fournir la fonction de hachage utilisée par des données fournies, pour peu qu'elles suivent une distribution indistinguable d'une distribution uniforme. Ainsi on convertit un attaquant qui réussi à forger une signature, en un adversaire qui va inverser la fonction trappe en question en un point parmi une suite de points donnés. Ce problème pour McEliece s'appelle R-SD dans [116].

Notons aussi que asymptotiquement parlant, inverser une fonction trappe en un point parmi une suite de points donnés, et l'inverser en un point donné sont aussi difficiles. Il n'en est pas de même en pratique, et souvent ce premier problème est en racine carré de la complexité du deuxième. Cela est la source d'impossibilité de faire de signatures courtes avec le même schéma de signature simple qui consiste à inverser la fonction trappe. Nous étudions ce problème en détails dans le chapitre 18.

2.4 La sécurité en chiffrement à clef publique

La cryptographie à clef publique, bien qu'inventée il y a 30 ans, continue à inquiéter. Beaucoup de familles de cryptosystèmes ont été cassées, pour peu qui restent, le moindre détail d'implémentation peut les rendre vulnérable. Des récents attaques réalistes sur l'utilisation de RSA, notamment dans le protocole SSL, il n'y a pas plus utilisé sur l'internet, ont montré une fois de plus l'extrême fragilité de la sécurité en cryptographie à clef publique.

Il faut d'abord remarquer que la sécurité d'un cryptosystème à clef publique, est beaucoup plus complexe que dans la cryptographie classique (symétrique). En effet, il y a énormément de nouveaux scénarios d'attaque qui n'existaient pas en cryptographie à clef secrète. Tout d'abord, tout adversaire à accès à un oracle de chiffrement (une boîte noire qui chiffre, donnée par définition même de la cryptographie à clef publique) qui, en plus, est donnée sous forme d'expression mathématique directe (ce qui est plus fort). Ce scénario est assez rare en cryptographie à clef secrète. On arrive même à avoir un accès, même partiel à un oracle qui déchiffre. Tout cela combiné intelligemment donne toute une classe d'attaques dits actifs, dans lesquels le fraudeur aura la possibilité d'interagir en temps réel avec les parties légitimes et obtenir des réponses à des questions qu'il ne pourrait jamais résoudre avec la seule connaissance des quantités publiques du schéma.

Le principaux attaques étudiés en cryptographie à clef publique sont dans l'ordre de généralité croissante :

1. L'attaque à clair choisi (chosen plaintext attack, CPA), le moins fort possible car toute personne peut chiffrer avec la clef publique.
2. L'attaque avec vérification du clair (plaintext checking attack, PCA), proposée en 2001 par Pointcheval et Okamoto.
3. L'attaque à chiffré choisi ((lunchtime or indifferent) chosen ciphertext attack, CCA1)
4. L'attaque à chiffré choisi adaptative (adaptive (chosen ciphertext) attack, CCA2), parfois appelé CCA tout court.

En apparence, les attaques actives (CCA2) ne semblent donner rien sur les quantités secrètes. Pourtant, la cryptographie à clef publique, très riche mathématiquement, abonde en réductions d'un problème à l'autre. Cela s'avère à double tranchant. D'un côté on arrive à faire des preuves sécurité relatives, et de l'autre côté cela permet d'extraire de quantités secrètes tout en posant des questions tout à fait anodines (mais bien choisies). Par exemple on arrive à signer un message avec RSA en demandant la signature d'un autre message de son choix, sans lien visible.

Faute de compréhension suffisante, on a à la fois redouté et sous-estimé, l'importance de ce type d'attaques pendant des décennies entières. Il a fallu comprendre le problème, et définir des notions de sécurité beaucoup plus générales, qui englobent toute sortes d'attaques adaptatives qui chercheraient à abuser des protocoles ou fonctions cryptographiques afin d'en extraire des informations secrètes. Dans le contexte de chiffrement, ce n'est qu'en 1998 que Bellare,

Desai, Pointcheval et Rogaway ont clarifié et ordonné toutes ces notions de sécurité [12].

Le notions les plus fondamentales sont dans l'ordre de généralité croissante :

1. Le sens unique (one-wayness, OW).
   Signifie l'impossibilité pour l'adversaire d'inverser la fonction de chiffrement en un point choisi au hasard. Cette notion de sécurité ne saurait être suffisante si par example l'ensemble de textes clairs et petit.
2. La sécurité sémantique (semantic security, polynomial security, IND)
   Elle est due à Goldwasser et Micali. Elle signifie l'impossibilité pour l'adversaire de distinguer entre le chiffrement de deux messages quelconques m_1 et m_2. Cette notion correspond à la sécurité parfaite, mais dans le contexte des adversaires dont les ressources sont bornés.
3. Non-malléabilité (Non-malleability, NM)
   Elle est due à Dolev, Dwork et Naor. Elle signifie l'impossibilité pour l'adversaire de produire étant donné un chiffré y, de produire un autre chiffré y' tels que les clairs correspondants x, x' satisfassent à une relation vérifiable. Par exemple $x = x' + 1$.

Il se trouve que dans le scénario d'attaque le plus général CCA2, la sécurité sémantique IND-CCA2 est équivalente à la non-malléabilité NM-CCA2, [12]. C'est la notion de sécurité la plus forte parmi celles que nous avons décrites. Elle englobe plusieurs notions plus faibles et assure la sécurité contre toutes sortes d'attaques dites actives. On l'appelle "chosen-ciphertext security".

La prise de conscience de ces notions de sécurité, vers la fin des années 90 avait amené des gens à chercher de cryptosystèmes pratiques qui soient effectivement sûrs contre ces attaques à chiffré choisi. Ainsi en 1998 Cramer et Shoup avaient proposé un tel système : à la fois pratique et prouvé sûr dans le sens quasiment le plus fort que connaît la science cryptologique. Il est en effet "chosen-ciphertext secure". L'ironie de l'histoire, ce résultat important va s'avérer presque superflu un an plus tard. En 1999 Fujisaki-Okamoto et indépendamment Pointcheval trouvent le moyen de transformer n'importe quelle fonction trappe, solide dans un sens très restrictif de sens unique, en un système prouvé sûr contre des attaques à chiffré choisi. En 2001 un nouvelle conversion appelée REACT, meilleure, plus simple et plus pratique est proposée par Pointcheval et Okamoto. REACT transforme toute fonction trappe sûre au sens faible OW-PCA en une fonction de chiffrement qui soit non seulement OW-CCA2, mais aussi IND-CCA2 et de façon équivalente NM-CCA2.

Il a fallu près de 30 ans après l'invention des premiers cryptosystèmes à clef publique pour enfin apprendre à les utiliser correctement. Désormais on pourra concentrer l'effort de recherche sur l'inversion des fonctions de trappe, et cela suffira pour avoir confiance en leur utilisation concrète dans la vie réelle.
L'application directe de ce résultat pourra être une utilisation prouvablement sûre du cryptosystème HFE. Il suffit désormais d'étudier le problème d'inversion de HFE. Ce problème est défini dans la section 12 et appelé le problème HFE, de même que le problème RSA est le problème d'inversion de RSA, est précisément étudié dans la présente thèse.

2.4.1 La conscience du texte clair

Il n'y pas de doute que des notions de sécurité encore plus fortes verront le jour, bien que pour l'instant il semble que "chosen-ciphertext security" est plus que satisfaisante pour assurer la sécurité réelle des cryptosystèmes.

Pourtant on connaît une notion **strictement** plus forte. C'est la conscience du texte clair (plaintext-awareness PA). Il s'agit d'une formulation très forte de la non-malléabilité, due à Bellare et Rogaway qui a par la suite été généralisée avec Pointcheval et Desai dans [12]. Il semblerait qu'il s'agit non pas d'une notion de sécurité, mais plutôt d'un outil de preuve qui permet de prouver NM-CCA2. dans la conscience du texte clair (PA) le but de l'attaquant est, en simplifié, de créer un chiffré valide y sans connaître à l'avance le clair correspondant x.

Actuellement il y a de problèmes de définition pour mettre en pratique cette notion semble-t-il intuitive. La conscience du texte clair (plaintext-awareness PA) n'est en effet définie pour l'instant que dans le modèle de l'oracle aléatoire. Plus précisément, on considère qu'un attaquant dispose de la clef publique, une fonction de hachage H (qui n'est accessible que sous forme d'un oracle) et d'un oracle de chiffrement qui inclut le hachage $x \mapsto \mathcal{E}_{pk}^{H}(x)$. Supposons qu'un adversaire B avait produit un texte chiffré y valide, i.e. $y = \mathcal{E}_{pk}^{H}(x)$ en interagissant avec H et $\mathcal{E}_{pk}^{H}$. Alors le système est PA, s'il est IND-CPA, et s'il existe un simulateur K (appelé **k***nowledge extractor*, ou *plaintext extractor*), qui en observant tous les chiffrés $c_i = \mathcal{E}_{pk}^{H}(p_i)$ obtenus par l'adversaire B, mais pas les clairs correspondants p_i, toutes les interactions avec H (les questions et les réponses) et sans avoir accès à l'oracle qui calcule $\mathcal{E}_{pk}^{H}$ ou H (!) arrive pourtant à extraire x avec probabilité proche de 1.

Dans le cadre du modèle de l'oracle aléatoire, PA est prouvée dans [12] **strictement** plus forte que NM-CCA2. Ce fait est par example utilisé pour prouver le sécurité de REACT [17]. Il n'est pas clair qu'un cryptosystème qui est NM-CCA2 sans être PA pose ou pas un quelconque problème de sécurité. Il semble donc inutile d'avoir un cryptosystème sûr au sens de PA. De plus, dans [12] les auteurs montrent que PA est caduque en dehors de l'oracle aléatoire. On note toutefois que tel que PA est défini, l'argument est tautologique et artificiel. Il y a en fait deux possibilités :

1. Ou bien PA n'a pas d'importance pour la sécurité d'un schéma à clef publique.
2. Ou bien, puisque sans une fonction à sens unique qui satisfait à l'hypothèse de l'oracle aléatoire il n'est pas possible d'avoir PA, cela veut dire que sans oracle aléatoire (i.e. sans outils de la cryptographie à clef secrète) on ne peut pas construire de cryptosystèmes à clef publique vraiment sûrs (PA), seulement dans le sens plus faible (NM-CCA2).

L'avenir montrera laquelle de deux versions est la bonne.

2.5 La sécurité des signatures

La sécurité prouvable dans le domaine de signatures est en général plus facile à obtenir que pour le chiffrement. La voie avait été tracée par les schémas Zero-knowledge connus depuis 1984, et qui apportent une sécurité prouvée contre tout type d'interaction. Malheureusement ces schémas sont conçus pour faire de l'authentification. Ils donnent aussi des signatures, comme expliqué dans 2.2.[6], mais assez longues en pratique.

Dans l'article [21] Stern et Pointcheval définissent précisément ce que un schéma de signature sûr. Le but de l'adversaire peut-être dans l'ordre de généralité décroissante :

1. Cassage total (total break) : récupérer la clef secrète. But trop faible pour être pris au sérieux, car bien de schémas sont cassés sans récupérer la clef secrète.
2. Falsification Universelle (universal forgery) : signer tout message.
3. Falsification Sélective (selective forgery) : signer certains messages.
4. Falsification Existentielle (existential forgery) : être capable de fabriquer une paire (message, signature) valide.

Le principaux attaques étudiés en signature sont dans l'ordre de généralité croissante :

1. L'attaque sans message (No-message attack), qui consiste à cryptanalyser la clef publique elle-même.
2. L'attaque à message connu (simple) ((plain) known-message attack).
3. L'attaque à message choisi générique (generic chosen-message attack). Les liste de messages à signer et choisie, mais avant de connaître la clef publique.
4. L'attaque à message choisi orienté (oriented chosen-message attack). Les liste de messages à signer et choisie après avoir connu la clef publique.
5. L'attaque à message choisi adaptative (adaptively chosen-message attack). Avoir accès à un oracle qui signe des messages de notre choix.

Le notion la plus puissante consiste donc à ne pas être capable de fabriquer une paire (message, signature) valide dans le contexte de l'attaque à message choisi adaptative. Cette façon de formaliser la sécurité de la signature électronique avait été introduite pour la première fois par Goldwasser, Micali et Rivest [19]. L'article [21] donne précisément des preuves de sécurité en ce sens fort, pour une large classe de schémas connus, et en particulier pour une variante de l'ElGamal. D'autres exemples abondent, par example ma preuve de sécurité de signatures avec McEliece dans [116].

2.6 La sécurité des schémas d'authentification

2.6.1 Authentification des personnes et les 3 facteurs

Une personne peut s'authentifier grâce à

1. une chose qu'elle est, p.ex. sa voix unique, ou empreintes digitales ;
2. une chose qu'elle sait, p.ex. un mot de passe ;
3. une chose qu'elle a, p.ex. une carte à puce, dongle USB, ou autre token.

Dans la vie réelle il est important de distinguer et combiner ces 3 facteurs. Les utiliser tous les trois est préférable que d'en utiliser un seul. On va s'intéresser uniquement au cas (3) : comment un token, qui est une machine peut s'authentifier à une autre machine. Il est difficile de croire que la machine elle-même soit difficile à reproduire avec du "reverse engineering", et donc (1) est impossible. Par contre une machine peut garder un secret beaucoup plus long qu'un simple mot de passe, qui doit être stocké dans un endroit protégé difficile d'accès (tamper resistance). Une machine peut également faire de calculs beaucoup plus complexes et implementer des schémas cryptographiques. Cela permet de ne pas divulguer directement le secret, uniquement des quantités dérivées.

2.6.2 Authentification des machines

Tout schéma de signature peut servir pour s'authentifier, il suffit de signer un challenge aléatoire. Réciproquement, il existe une classe de schémas d'authentification particulièrement forte, appelée les schémas à divulgation nulle de connaissance (Zéro-knowledge), et que nous introduisons en détails dans 25.4.0.1, qui permet de construire des schémas de signature, comme nous l'avons déjà expliqué dans 2.2.[6].

Un schéma d'authentification contient deux parties : le Prouveur et le Vérifieur. Il s'agit de prouver son identité, ou du moins en convaincre. Certains chercheurs appellent des arguments de preuves qui nécessitent des hypothèses supplémentaires de type Oracle Aléatoire (ROM) [14, 140, 141, 142, 146].

Il existe de nombreux autres schémas d'authentification, notamment utilisant les fonctions de hachage, le chiffrement symétrique, ou les codes spéciaux (authentication codes). Ils peuvent assurer une excellente sécurité, parfois inconditionnelle, mais protègent contre les adversaires autrement moins puissants : passifs et qui ne peuvent pas corrompre des parties.

Seuls les schémas à clef publique peuvent protéger contre un adversaire qui est capable de corrompre (p.ex. désassembler) un Vérifieur pour en extraire le secret nécessaire, en espérant de se faire passer pour le Prouveur. De plus, si l'on veut se protéger des adversaires actifs, qui peuvent poser des questions de façon adaptative afin d'extraire des informations, alors seuls des schémas à divulgation nulle de connaissance (Zéro-knowledge) permettent de **prouver** la sécurité dans un tel scénario.

2.7 Des principaux schémas asymétriques connus

On va esquisser une classification des principaux schémas qui existent à ce jour dans la cryptographie à clef publique.

1. Fonctions trappe déterministes
   (a) Univariables :
      i. sur $\mathbb{Z}/N\mathbb{Z}$, N grand - RSA, Rabin
      ii. sur de groupes plus complexes - Courbes Elliptiques
      iii. sur de petits corps finis - Matsumoto-Imai (C^*), D^*, [C] (HM), HFE, Cade
   (b) Multivariables Linéaires :
      i. sur $\mathbb{Z}$ - sacs à dos de Merkle-Hellman, réduction des réseaux.
      ii. sur des (petits) corps finis - McEliece, Niederreiter, GPT et les variantes,sacs à dos de Chor-Rivest.
   (c) Multivariables Quadratiques :
      i. sur $\mathbb{Z}/N\mathbb{Z}$, N grand - Ong-Schnorr-Shamir, birational permutations de Shamir.
      ii. sur des (petits) corps finis - Matsumoto-Imai (C^*), D^*, [C] (HM), HFE, UOV, HFEv-, TPM , TTM, Flash, Sflash, Quartz.
2. Schémas d'authentification à divulgation nulle de connaissance (Zéro-knowledge)
   (a) Univariables :
      i. sur $\mathbb{Z}/N\mathbb{Z}$, N grand - Fiat-Shamir, GQ, GQ2
   (b) Multivariables :
      i. sur $\mathbb{Z}$, PPP
      ii. sur des petits corps finis - PKP, SD, CLE, Chen, MinRank (présente thèse), IP, GI
3. Schémas de signature - peuvent être construits à partir de tous les schémas de (1) ou (2).
4. Schémas à clef publique probabilistes - construits à partir des schémas de (1) mais aussi bcp. d'autres.

Remarque sur les schémas de signature Les schémas de signature à base des fonctions trappe déterministes (1) sont plus pratiques, mais les schémas à base de Zéro-knowledge (2) donnent davantage de preuves de sécurité.

Notamment, on connaît 4 schémas SD, PPP, PKP et MinRank dont la sécurité est basée sur un problème NP-complet, dont un (MinRank) fait partie des contributions de la présente thèse. De nombreux schémas de signature basées sur (1) et pratiques ont été prouvés sûrs, voir 2.5.

Remarque sur les schémas probabilistes Certains schémas récents peuvent atteindre des notions de sécurité (notamment contre les attaques actives) beaucoup plus élevées que ne peut atteindre un schéma déterministe : sécurité sémantique, non-malléabilité, décrites dans 2.4, ainsi que d'autres encore plus forte telles que la sécurité du chiffrement multiple etc. Cependant n'importe quel cryptosystème déterministe (1) peut être transformé en schéma probabiliste avec un bon protocole utilisant un aléa et une fonction de hachage, et ainsi résistant en pratique à toutes sortes d'attaques. Pour ne pas se tromper certaines de ces conversions ont été prouvé sûres très récemment, notamment REACT [17]. Tout n'est pas encore résolu, mais il semble suffisant de disposer des fonctions trappe non-inversibles dont plusieurs exemples tel HFE sont étudiés dans le présente thèse, pour obtenir des cryptosystèmes prouvés sûrs, voir 2.4.

Chapitre 3

La cryptographie multivariable

Tous les mathématiciens savent
que le passage de une à plusieurs variables
est un "saut" brusque,
qui s'accompagne de grandes difficultés
et nécessite des méthodes toutes nouvelles.
Jean Dieudonné

3.1 Entre la cryptographie univariable et multivariable

La cryptographie à clef publique la plus utilisée à ce jour (voir aussi 2.7) peut être qualifiée de univariable (parfois bivariable). On y rencontre les plus souvent des équations **univariables**, essentiellement l'exponentiation, dans les anneaux ou corps finis. La structure algébrique d'anneau, avec l'interaction de l'addition et multiplication, s'avère suffisamment riche pour mener à des attaques sous-exponentielles. Cela oblige à utiliser des très larges blocks, par example de 1024 bits, pour obtenir un niveau de sécurité acceptable. On peut aussi dire que les cryptosystèmes basées sur les courbes elliptiques (EC) utilisent une équation bivariable. Dans ces cryptosystèmes on n'obtient plus qu'un groupe abélien, et le seuls attaques actuellement connues sont exponentiels. Il y a toutefois toujours une attaque en racine carré de la recherche exhaustive, qui découle du fait que c'est un groupe, et la taille de bloc minium est ainsi de $2 \times 80 = 160$ bits.

Dans la cryptographie multivariable, on utilise plusieurs équations avec plusieurs variables sur un corps ou anneau fini. Assez souvent il n'y a plus de structure de groupe sur l'ensemble, ni aucune autre structure algébrique apparente. Cela permet de concevoir des cryptosystèmes qui ne sont pas cassés, même en disposant d'une puissance de calcul exponentielle, et au delà de la racine carré de la recherche exhaustive. Cette propriété remarquable nous permettra par la suite de concevoir des schémas qui manipulent de blocs très courts (80 ou 100 bits) et qui donnent des signatures très courtes.

3.1.1 Classification des schémas multivariables

La cryptographie multivariable est une branche de la cryptographie à clef publique basée sur des systèmes d'équations polynomiales sur des petits corps/anneaux finis. On y distingue trois principaux courants, dont la présente thèse représente tous. Ce sont (voir aussi 2.7) :

1. Des cryptosystèmes multivariables linéaires sur des petits corps finis Dans 2.7 il sont classés dans (1.b.iii) et (2.b.ii).

2. Des schémas multivariables quadratiques sur des petits corps finis Dans 2.7 il sont classés dans (1.c.iii). Certains de ces schémas apparaissent aussi dans (1.a.iii), car on peut les voir aussi bien dans une représentation univariable, que multivariable. Il est intéressant de remarquer que c'est précisément la possibilité d'une représentation univariable qui est source d'attaques :

- Attaques de C^* par Patarin [56, 8] et 13.3.6.
- Attaques de 'basic' HFE par Shamir-Kipnis [71] et 13.2.
- Attaques sur D^* par Courtois [58, 8].
- Attaques sur HM par Courtois [58, 8].

Cela explique l'intérêt de :

3. Les variantes combinatoires de ces schémas. Si on enlève à un des quatre schémas mentionnés ci-dessus (C^*, HFE, D^*, HM), un certain nombre $r > 10$ d'équations dans leur représentation multivariable, ils perdent totalement leur représentation univariable. On obtient alors des schémas appelés respectivement C^{*--}, HFE^{--}, D^{*--}, HM^{--}, pour lesquels on ne connaît aucune attaque, même théorique. En fait aucune attaque n'est actuellement connue pour des schémas **purement multivariables** avec les paramètres bien choisis, voir 7.6.

3.1.2 Le principe directeur

Le design des schémas multivariables cherche d'abord à trouver des schémas **algébriques** de base, qui soient reliés ou directement basés sur des problèmes algébriques difficiles à résoudre. Ensuite on ajoute une ou plusieurs couches **combinatoires** (voir 3.2.3) qui introduisent des perturbations mettant à mal le peu d'attaques qui existent contre le sous ensemble algébrique interne.

3.2 La sécurité des schémas multivariables

3.2.1 Des schémas faibles

Certains schémas ont surpris par la simplicité de l'attaque, par exemple la cryptanalyse de C^* par Patarin [56], ou celle de HM par Courtois [58]. Cela incite de gens à rester méfiant.

3.2.2 Des problèmes algébriques difficiles

Cependant on connaît aujourd'hui des schémas dont la sécurité est reliée à des problèmes difficiles et connus : notamment le problème HFE, qui lui-même est lié à trois autres problèmes algébriques MinRank, MQ et IP. Ces quatre problèmes sont étudiés en détail dans la présente thèse, avec des contributions importantes dans l'état de l'art sur chacun d'eux. Ils se sont avérés d'être tous de problèmes difficiles, voilà un résumé des résultats de la présente thèse :

1. IP est étudié dans 19. Nous avons montré qu'il n'est pas NP-dur dans [88]. Dans [58, 88] on a pu trouver une attaque en $q^{n/2}$ au lieu de q^{n^2} mais cela reste exponentiel.
2. MQ auquel est consacrée la partie 6, est NP-dur, et si le nombre d'équations est égal au nombre de variables, le meilleur algorithme connu est souvent la recherche exhaustive [42].
3. MinRank auquel est consacrée la partie 21, est NP-complet. Il s'est avéré être un problème très difficile, car il contient de nombreux problèmes très célèbres de cryptographie, voir 23 et 23.1. C'est peut-être le plus difficile de tous les problèmes algébriques jamais rencontrés en cryptographie.
4. HFE défini dans 12 s'est avéré sous-exponentiel mais difficile en pratique si les paramètres sont bien choisis. De plus, les attaques échouent pour les variantes de HFE 17.1. Les meilleures attaques connues sur HFE, publiées en partie dans [68] sont décrites dans de la présente thèse.

3.2.3 Versions combinatoires - renforcement de la sécurité

Pour tout cryptosystème multivariable on peut, en partant du schéma de base, en proposer de nombreuses variantes. Il existe un certain nombre d'opérations, devenues quasiment standard, qui sont des astuces permettent de transformer un schéma cryptographique multivariable en un autre. Cela donne des performances légèrement dégradées, mais cela renforce toujours le caractère multivariable du schéma en s'approchant d'avantage d'un ensemble d'équations quadratiques (MQ) totalement aléatoire. Les quatre transformations de base sont, en utilisant les notations de Jacques Patarin [7, 65] :

Le $-$: enlever des équations de sorte à perdre des propriétés globales du système d'équations, notamment la représentation univariable.

Le v : ajouter de nouvelles variables dont la place est cachée , et qui permettent (même en clef secrète) de n'inverser la fonction **que** si leurs valeurs sont fixées.

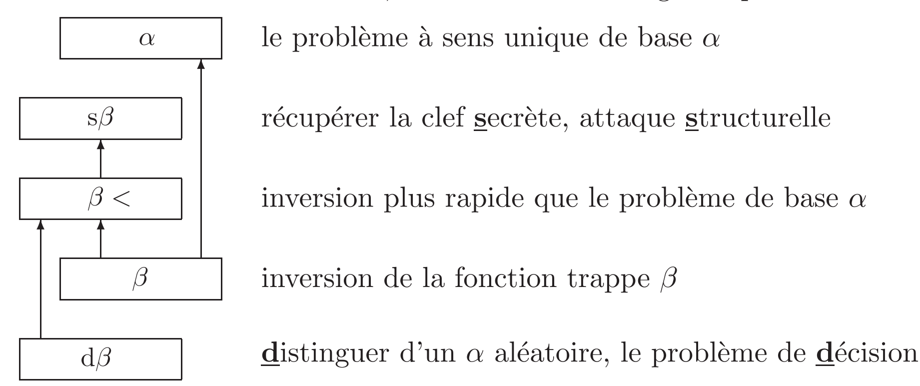
Le f : fixer des variables de sorte à perdre des propriétés globales du système d'équations, notamment la représentation univariable.

Le $+$: ajouter des équations de sorte à noyer des propriétés globales du système d'équations, notamment la représentation univariable.

Parfois la définition exacte de ces opérations varie d'un cryptosystème à l'autre. Les transformations peuvent être combinées pour créer une infinité de variantes possibles. Pour plus de détails et les applications, notamment à HFE, voir 17.1 et aussi [7, 8, 65].

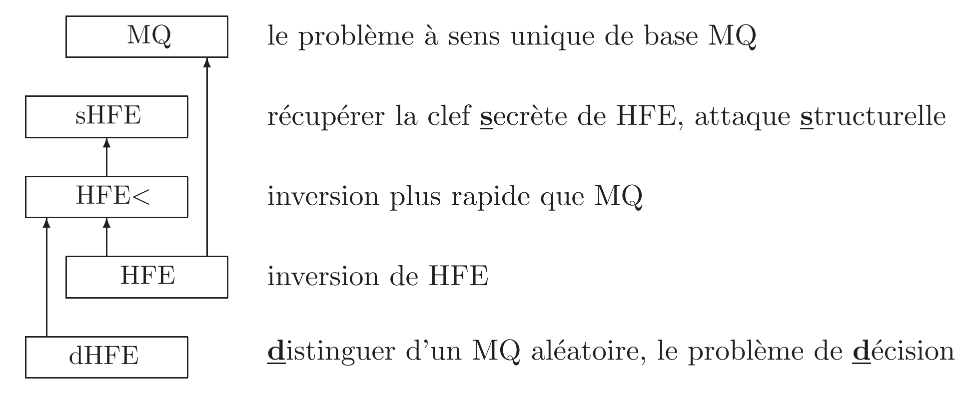
3.3 La hiérarchie des problèmes

Une fonction trappe β est d'habitude construite en partant d'une fonction à sens unique donnée α, en y cachant une structure algébrique ou combinatoire. Si on appelle α est le problème qui consiste à inverser cette fonction et β est le problème qui consiste à calculer les inverses pour la fonction trappe β, il se pose naturellement un certain nombre de problèmes annexes. Ces problèmes sont décrits sur le schéma suivant, avec des réductions génériques :



Les problèmes fondamentaux liés à une fonction trappe β

Par exemple pour une fonction trappe multivariable quadratique, le problème de base est le problème α=MQ, défini dans 6 qui consiste à résoudre un certain nombre d'équations **Q**uadratiques **M**ultivariables sur un corps fini. Le cryptosystème HFE défini et étudié dans la partie IV et dont le "véhicule" est précisément MQ, donne ainsi lieu aux problèmes suivants :



Les problèmes fondamentaux liés à HFE

Le schéma de base ci-dessus se reproduit pratiquement à l'identique pour de nombreux autres cryptosystèmes, y compris ceux qui ne sont pas multivariables.

En plus des réductions génériques décrites sur ce schéma, on a souvent des réductions entre des problèmes qui viennent des cryptosystèmes différents, ou de nombreuses versions d'un même schéma. Sur le schéma qui suit, on resume des liens et des réductions connue entre de très nombreux problèmes qui apparaissent en cryptographie multivariable, tout au long de ce document et dans la littérature :

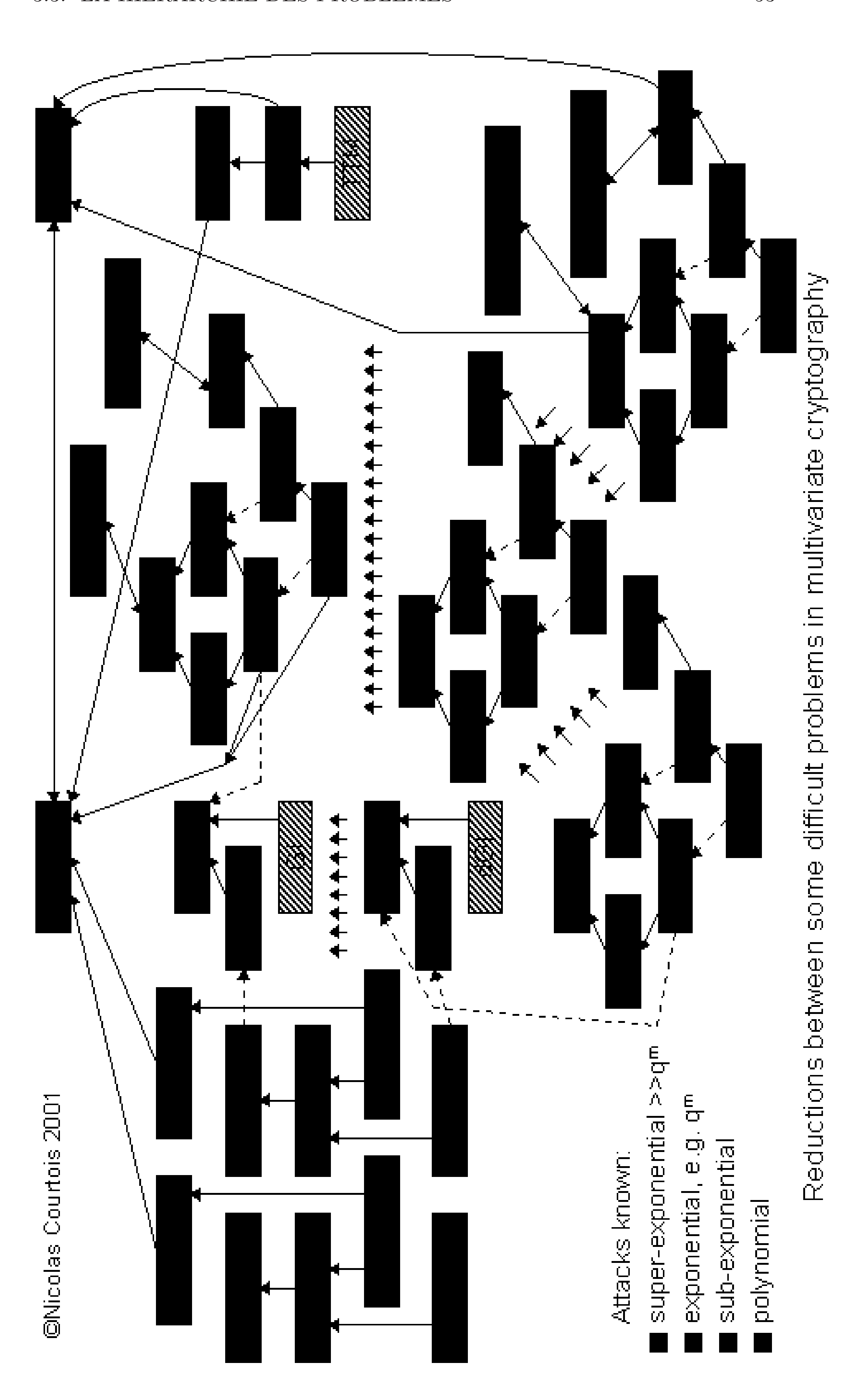


FIG. 3.1 – Les problèmes connus en cryptographie multivariable avec réductions

Chapitre 4

Solutions classiques vs. multivariables

4.1 Les avantages comparatifs

On va essayer de placer les principaux schémas multivariables dans le contexte des schémas à clef publique bien connus, tels que RSA, McEliece ou Courbes Elliptiques. Le plus connu d'entre aux, le RSA, demeure depuis plus de 24 ans, la référence numéro un de la cryptographie à clef publique.

4.1.1 Vers la fin du règne de RSA

Deux événements récents largement médiatisés ont attiré l'attention du public sur la taille des modulus RSA :

1. Le pirate Français, Serge Humpich, a désassemblé un terminal de paiement pour trouver la clef publique RSA utilisée par les banques pour signer les cartes bancaires. Ainsi il a pu fabriquer de fausses cartes capable de se faire passer pour des vraies dans certains terminaux simplifiés.

   Pour cela, Serge Humpich a cassé RSA en factorisant un entier RSA de 320 bits. Pourtant, au moment du déploiement de la carte bancaire en France dans les années 80, la taille de 320 bits avait été jugée (à tort) largement satisfaisante.
2. Mieux encore, en août 1999 avec un effort conjoint de chercheurs de nombreux pays, dont l'équipe de l'École Polytechnique avec François Morain, on a pu casser le challenge RSA de 512 bits.

Ainsi, la demande de trouver un digne successeur à RSA est devenue d'actualité. Les tailles de paramètres qui assurent la sécurité de RSA deviennent de plus en plus encombrantes, par exemple 1024 bits. On peut difficilement considérer cela pratique et raisonnable.

4.1.2 Les cryptosystèmes asymétriques à blocs courts

Dans les paragraphes qui suivent nous débattront des avantages et des inconvénients de trois candidats majeurs : Les Courbes Elliptiques, McEliece et

HFE, qui sont les seuls candidats crédibles qui donnent aujourd'hui les tailles de blocs (et tailles de signatures) **très petites** à l'encontre de RSA.

4.1.3 Quelle Alternative pour RSA ?

RSA est basé sur une équation modulaire avec une variable,
Une généralisation naturelle est de considérer des systèmes
de plusieurs équations avec plusieurs variables (...)
HFE est considéré comme un des plus forts cryptosystèmes de ce type. (...)
ADI SHAMIR

Dans un certain sens aussi bien HFE que les Courbes Elliptiques (EC) sont des voies de généralisation de l'idée RSA :

EC Utiliser des groupes plus complexes :
Les courbes elliptiques [Koblitz, Miller, Crypto'85], [33, 32].

HFE Utiliser des polynômes plus complexes :
HFE [Patarin, Eurocrypt'96], [65].

Le raison pour laquelle les progrès de cryptanalyse de RSA ont été aussi rapides, est que la structure algébrique de $\mathbb{Z}/N\mathbb{Z}$ est trop riche. Le problème d'extraction des racines dans $\mathbb{Z}/N\mathbb{Z}$, appelé *le problème RSA*, est malheureusement un problème sous-exponentiel.Il serait don intéressant de disposer de...

Les cryptosystèmes exponentiels ?

Il y a seulement deux tels cryptosystèmes bien établis, qui permettent de faire à la fois de la signature et du chiffrement :

McEl Le système de McEliece, connu depuis 1978 [127]. Ce n'est qu'en 2001 que Matthieu Finiasz, Nicolas Sendrier et moi-même avons montré (plusieurs façons) de faire des signatures avec, voir [116].

EC Les courbes elliptiques, connues dans leur application cryptographique depuis 1985 [Koblitz, Miller], [33, 32].

Malheureusement dans les deux cas, et grâce à l'existence de différents homomorphismes de groupe, il existe toujours une attaque en $\sqrt{\text{recherche exhaustive}}$.

Peut-on faire mieux ? Le problème est ouvert. Actuellement il n'y a **que** des divers cryptosystèmes multivariables, qui semblent apporter une sécurité proche de la recherche exhaustive (!). Il s'agit ici de la sécurité en pratique, pour des paramètres fixés, car paradoxalement il y a des doutes sur le caractère réellement exponentiel de ces schémas et leur sécurité contient encore beaucoup d'inconnues. Seuls les candidats de la famille HFE rassurent, de par la diversité des problèmes difficiles qui restent à résoudre pour les cryptanalyser.

4.1.4 Fondements théoriques de la sécurité.

RSA - basé sur un problème algébrique, la factorisation
- le problème d'inversion lui même est appelé le problème RSA et semble aussi que dur que la factorisation.

McEl. - ils reposent sur l'observation que les codes de Goppa sont assez nombreux et assez optimaux pour ne pas savoir les distinguer d'un code aléatoire.
- le problème de Syndrome Decoding SD (pour un code aléatoire) est conjecturé pleinement exponentiel à résoudre.

EC Les courbes elliptiques [33, 32] généralisent les problèmes DL et RSA sur quelque chose de 'très compliqué'. Ils ne se basent **que** sur l'opacité de la représentation du groupe sous-jacent, aucune attaque n'existe qui utiliserait autre chose que le fait que c'est un groupe (du moins pour l'instant).

HFE La sécurité se décline à deux niveaux emboîtés. Les deux sortes de sécurité : problèmes algébriques et l'opacité de la représentation sont présents simultanément dans HFE :

(a) Algèbre : Des problèmes difficiles **MQ, IP, HFE, MinRank**.

(b) Les modifications des schémas : Détruisent la structure algébrique. Des opérations $+$, $-$,v, f (décrites dans 3.2.3 et 17.1).
Même si l'opacité (de ces opérations) est percée par une cryptanalyse future, ce sera toujours au moins aussi difficile à casser que le problème algébrique HFE.

4.1.5 La sécurité en pratique.

McEl. Les paramètres d'origine $(n, k, d) = (\mathbf{1024}, 524, 101)$ ont été cryptanalysés en 1998 par Anne Canteaut. L'attaque demande environ 2^{60} opérations CPU.

RSA **512 bits** avait été cassé en 08.1999 et l'attaque avait été présentée à Eurocrypt 1999. Elle nécessite 8 000 MIPS-années, c'est à dire environ 2^{58} opérations CPU.

EC **97 bits** - le challenge de Certicom.com avait été cassé en 09.1999. L'attaque nécessite 15 000 MIPS-années, c'est à dire environ 2^{59} opérations CPU.

HFE (a) Basic HFE **80 bits** (connu en tant que HFE Challenge 1) - La meilleure attaque connue en 2^{62} que trouvée par l'auteur de la présente thèse en Décembre 1998. Elle est décrite dans le chapitre 15.1.3 de la présente thèse et dans [68].

(b) Pour les modifications de HFE **80 bits**, par exemple HFE–, HFEv, HFEv- [65, 70, 68, 7], la meilleure attaque connue reste l'attaque générique sur MQ, toujours proche de la recherche exhaustive. On n'arrive pas à exploiter ni même à détecter l'existence de la trappe dans HFE.

On constate qu'à difficulté comparable, les cryptosystèmes multivariables semblent atteindre de tailles de blocs non pas de $1024, 512, 97$ bits, mais aussi courts que 80 bits.

4.2 L'importance de la cryptographie multivariable

L'apport de la cryptographie multivariable à la cryptographie à clef publique en général est de multiple nature.

4.2.1 Plus vite, mieux, moins cher

Tout d'abord, les schémas étudiés sont souvent beaucoup plus rapides, plus intuitifs, et donnent des signatures beaucoup plus courtes que des schémas concurrents.

4.2.2 Diversification des bases

La sécurité na saurait reposer sur un même problème difficile tel que la factorisation. On a besoin de "biodiversité" de problèmes et d'instances, terme forgé par François Morain. L'idéal est bien entendu de faire mieux que la factorisation. Certains cryptosystèmes multivariables semblent reposer des bases théoriques **plus saines** que la factorisation : les problèmes NP-complets. Certains d'entre eux semblent en plus difficiles aussi en moyenne et semblent être des problèmes pleinement exponentiels, notamment les problèmes SD et MinRank.

4.2.3 Propriétés uniques

La cryptographie multivariable a des propriétés uniques et permet de choses qu'aucune autre branche ne permet. Par exemple elle permet de faire des signatures très courtes, étudiés en détails dans le chapitre 18. Elle m'a également permis de concevoir un schéma assez surprenant, basé sur HFE, dans lequel la clef publique est générée par l'autorité, alors que la clef secrète est générée de façon sûre chez soi [69]. Le but de ce schéma est de résoudre le problème réputé très difficile, d'offrir une infrastructure de communication avec de la signature et de l'authentification, mais qui ne donne pas aux utilisateurs de possibilité de faire du chiffrement sans volonté expresse de deux parties (sinon c'est inévitable). Voir [69].

4.3 Les applications industrielles

Avec les algorithmes HFE et MinRank, et aussi la variante économique de HFE- qui est Flash [61, 62] et Sflash (voir 9.6.3), on peut implementer n'importe quelle infrastructure à clef publique (chiffrement, authentification, signature) de façon particulièrement économe en ressources, par exemple dans les cartes à puce.

4.3.1 Les applications du MinRank

La contribution majeure de la thèse, c'est le nouvel algorithme à divulgation nulle de connaissance (Zéro-knowledge) MinRank qui fait de la signature numérique et de l'authentification. MinRank est basée sur le célèbre problème NP-complet MinRank qui généralise de nombreux problèmes célèbres et très difficiles (voir 23 et 23.1).

Il n'y a que très peu de schémas proposés qui reposent comme MinRank, sur un problème NP-complet : Le schémas PKP de Shamir [152], le schéma PPP de David Pointcheval [149], et les schémas de Stern et Chen [137, 139, 130, 131]. Parmi ces schémas, comme nous allons le voir dans 26.1.2, les plus performants sont MinRank, PKP et CLE.

4.3.2 Les applications de HFE

Par la suite de travaux récents, dont la meilleure attaque connue de HFE décrite dans la présente thèse et en partie dans [68, 70] il semble que le cryptosystème HFE est assez solide. Il est difficile de prévoir quelle sera l'évolution exacte des attaques sur HFE, mais il est clair qu'il offre une certaine sécurité et on peut difficilement imaginer des progrès spectaculaires. Casser HFE n'est probablement pas polynomial.

Le problème numéro un de la cryptologie appliquée, est sans aucun doute le problème de la signature numérique. Tous ce qui est fait dans la présente thèse concerne des signatures numériques.

Le cryptosystème HFE permet de faire de signatures numériques particulièrement courtes (80-128 bits) étudiées dans 18 et [65, 7]. Le seul autre système qui en est capable est le très récent schéma de signature basé sur McEliece [127, 128], proposé par Matthieu Finiasz, Nicolas Sendrier et moi-même [116]. HFE et McEliece sont les seuls cryptosystèmes connus qui permettent d'obtenir de signatures de moins de 100 bits.

Schéma	longueur		
RSA	1024	bits	
Courbes Elliptiques	321	bits	
DSA	320	bits	
HFEv-,Quartz	120-128	bits	voir 18.4.3
HFEf+	92	bits	voir 18.4.2
McEliece	87	bits	voir [116]

Table 4.3.2.1. Des longueurs des signatures avec une sécurité de 2^{80}

Troisième partie

Le problème MQ

Chapitre 5

Les corps finis en cryptographie

5.1 Rappels sur des corps finis et leurs extensions

Un corps est un anneau unifère K tel que tous les éléments de $K^* = K - \{0\}$ sont inversibles.

Dans de nombreux langues un corps est toujours commutatif, par example en anglais (*field) ou en polonais (*ciało). En français corps est une notion plus générale. Il faut s'en rappeler pour éviter des confusions. En revanche cela ne fait aucune différence pour les corps finis, car un théorème très connu [du à J.H.M. Wedderburn] affirme que tout corps fini est commutatif.

Du point de vue de l'informatique, informel, on peut dire qu'un corps est une structure de données qui permet d'effectuer les 4 opérations $+, -, \times, /$ à l'exception de division par zéro, et qui est gouverné par les lois de l'algèbre usuels, par exemple la distributivité :

$$(a+b)c = ac + bc$$

La seule particularité des corps finis par rapport à des exemples de corps connus tels que $\mathbb{Q}$, $\mathbb{R}, \mathbb{C}$, est que leur caractéristique est finie, ce qui veut dire que l'addition est périodique de période p, qui est toujours un nombre premier.

$$\underbrace{1 + 1 + \ldots + 1}_{\text{p fois}} = 0$$

Cette propriété implique en particulier pour $p = 2$ que l'addition et la soustraction coïncident. L'addition dans un corps de caractéristique 2 est une addition modulo 2 composant par composant, souvent implementée très efficacement par un XOR.

5.1.1 Classification des corps finis

Soit K - corps fini.

Exemples :

1. $K = \mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z} = \{0, 1, + \bmod 2, \times \bmod 2\}$.

2. $K = \mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ - des entiers modulo p, un nombre premier.
3. $K = \mathbb{F}_q$, $q = p^\alpha$.

Tous les corps finis sont sous forme $\mathbb{F}_{p^\alpha}$, cela bien évidemment à l'isomorphisme près. Ainsi, pour un corps fini ayant q éléments on écrira $K = \mathbb{F}_q$, même s'il s'agit seulement d'une identification à l'isomorphisme près, et que trouver cet isomorphisme n'est pas toujours facile.

Dans la littérature anglophone on écrit plutôt $K = GF(q)$ GF=Galois Field, du nom du mathématicien français de génie, Évariste Galois).

5.1.2 Propriétés de base

Le groupe multiplicatif $\mathbb{F}^*_{p^\alpha}$ d'un corps fini est cyclique. Il en découle que tout élément d'un corps fini $\mathbb{F}_{p^\alpha}$ satisfait l'équation suivante appelée l'équation du corps :

$$X^{p^\alpha} = X.$$

Soit f le polynôme $f = X^{p^\alpha} - X$.

Tout corps fini $\mathbb{F}_{p^\alpha}$ est isomorphe au corps de décomposition sur $\mathbb{F}_p$ du polynôme $f = X^{p^\alpha} - X$. Ce corps de décomposition peut être constitué par exemple, par l'ensemble des racines de $X^{p^\alpha} = X$ dans la clôture algébrique $\overline{\mathbb{F}_{p^\alpha}}$.

D'ailleurs il suffit pour cela de disposer d'une extension algébrique quelconque de $\mathbb{F}_p$ qui contient une racine $\lambda \notin \{0, 1\}$ de f. Elle le contient alors toutes et ce sont les $\{\lambda^i\}_{i=0,\ldots,p^\alpha-1}$. Une façon assez courante de construire une telle extension est de considérer des anneaux de polynômes (polynomial rings).

5.1.3 La construction des corps finis

On a vu que le corps $\mathbb{F}_p$, p premier, est simplement l'ensemble $\mathbb{Z}/p\mathbb{Z}$ des entiers modulo p.

La construction du corps $K = \mathbb{F}_{p^\alpha}$

- $\mathbb{F}_p[X]$ = est l'ensemble de polynômes en X avec les coefficients pris modulo p.
- Soit P un polynôme irréductible de degré α sur $\mathbb{F}_p$.
- $\mathbb{F}_{p^\alpha} \stackrel{def}{=} \mathbb{F}_p[X]/P(X)$, c'est l'ensemble des polynômes de $\mathbb{F}_p[X]$ modulo $P(X)$.
- $\mathbb{F}_{p^\alpha}$ est une extension du corps (de base) $\mathbb{F}_p$. C'est aussi un espace vectoriel de dimension α sur $\mathbb{F}_p$:
  tout élément $x \in \mathbb{F}_{q^\alpha}$ peut être codé comme les α coefficients d'un polynôme $x \in K[X]/P(X)$.

À partir de maintenant on supposera toujours que l'on travaille sur un corps fini de base $K = \mathbb{F}_q = \mathbb{F}_{p^\alpha}$ de caractéristique p.

De façon identique à celle décrite ci-dessus, on peut construire des extensions successives de K, voir le chapitre 12 concernant HFE.

5.2 Pourquoi les corps finis en cryptographie

Les corps finis sont aujourd'hui largement utilisés en cryptographie, aussi bien pour HFE, McEliece, les courbes elliptiques et beaucoup d'autres dont certains ont été cités dans 2.7 que pour de différents algorithmes d'authentification comme MinRank décrit dans 25.2 ou ceux basés sur les codes correcteurs ou [135, 134, 137, 139, 131].

Il doit y avoir une raison algébrique à cela.

Et pourquoi les grands premiers ? Une des raisons pour lesquelles on utilise les grands nombres premiers en cryptographie est la suivante.

Si l'on a un système d'équations à résoudre modulo N, N grand, pour tout $p|N$ il existe un morphisme (évident) de $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ qui permet d'obtenir des équations plus simples mod p, avec des tailles de données plus petites.

Autrement dit si l'on mesure la complexité d'une équation par la taille des données, on obtient de nouvelles équations non-triviales et strictement plus simples, ce qui contredit notre idée de fonction à sens unique expliquée dans 10.3.2.

Ainsi si N est le produit de très grands nombres premiers, il est impossible d'obtenir des équations plus simples sans factoriser N.

En d'autres termes, il n'y pas de morphisme $\mathbb{Z}/N\mathbb{Z} \to A$ avec A petit (ou qui soit autrement facile à trouver).

Pour les corps finis on a aussi une propriété analogue :

Théorème : Il n'y a pas de morphisme $\mathbb{F}_q \to K$ dans aucun corps plus petit K.

Notons que les attaques sur les cryptosystèmes basés sur les courbes elliptiques visent également à construire un morphisme vers un sous groupe d'un groupe plus simple (p.ex. un groupe cyclique) qui devrait avoir la propriété de ne pas être trop grand. De même un cryptosystème proposé à Crypto'2000, XTR, est basé sur le fait que le seul sous groupe cyclique dans lequel on sait plonger le groupe utilisé, est assez grand.

Chapitre 6

Étude du problème MQ

6.1 Le problème MQ (Multivariate Quadratic)

6.1.1 Le problème MQ, définition générale :

Équations **Q**uadratiques **M**ultivariables.

Soit K un anneau.
Soit f une fonction définie comme **m** polynômes quadratiques (pas forcément homogènes), avec **n** variables sur K.

$$f : \begin{cases} f_k(a_1, \ldots, a_n) & = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk}\ a_i a_j \\ \text{avec } k = 1..m, & \quad a_0 = 1 \end{cases}$$

Le problème $\mathbf{MQ_K}$

Soit $b = (b_1, \ldots, b_n)$ une sortie donnée.
Trouver (au moins) une solution $a = (a_1, \ldots, a_n)$ telle que

$$f(a) = b.$$

6.2 Quelques cas limites de MQ

6.2.1 MQ avec $n = m = 1$

Quand $n = m = 1$ on a deux cas assez différents :

- $\mathbf{K} = \mathbb{Z}_N$ - MQ équivalent à la factorisation de N (le chiffrement de Rabin), et donc probablement pas NP-complet (le théorème de Brassard [10, 7]).
- $\mathbf{K} = GF(q)$ - MQ est résolu non seulement pour les équations de degré $d = 2$, mais aussi pour tout degré d fixé par des algorithmes de factorisation de polynômes univariables connus depuis Berlekamp [1967], voir [3, 65, 70]).

6.2.2 D'autres MQ avec n, m petits

- **K** $= \mathbb{Z}_N$ Quand $n = 2$ $m = 1$ MQ n'est plus aussi solide que la factorisation, et ainsi le schéma de Ong-Schnorr-Shamir qui utilise une équation de ce type avait été cassé [4].
  Le cas $n = 1$ $m = 2$ n'est pas dur non plus, car avec deux équations quadratiques univariables on obtient une équation linéaire.
- **K** $= GF(q)$ - Quand $n = 1$ les méthodes univariables mentionnées ci-dessus s'appliqueront. Sinon, on peut espérer de fixer tous les variables sauf une et résoudre. Ainsi on résoudra le cas $m = 1$, $n = 2$.

6.2.3 MQ avec $m << n$

Si $m < n$ il y a en moyenne q^{m-n} solutions.

On peut en trouver une (ou un petit nombre), en fixant au hasard $n - m$ variables et en se ramenant à m équations avec m inconnues, le cas central de MQ largement étudié dans la partie suivante, voir 7.5.4 pour les conclusions.

Dans le chapitre 8 nous montrons des algorithmes plus efficaces pour traiter ce cas. Leur complexité tend vers $q^{m/2}$.

Dans l'article [76], on apprend comment trouver une solution à un système avec $n \geq m^2$ en temps polynomial sur un corps de caractéristique 2.

Le problème de faire la même chose sur un corps de caractéristique > 2 reste ouvert.

6.2.4 MQ avec $m >> n$

Si $m \geq n^2/2$ alors MQ est facile à résoudre par la méthode de linéarisation comme suit :

Linéarisation [méthode connue] :
- On prend approximativement $n^2/2$ nouvelles variables $y_{ij} = x_i x_j$.
- On obtient m équations **linéaires** avec $n^2/2$ variables.
- Puisque $m \geq n^2/2$ on fait la réduction de Gauss.

Ainsi le problème MQ est facile quand $m = \varepsilon n^2$ avec $\varepsilon = 1/2$. Que se passe-t-il quand $\varepsilon < 1/2$?

6.3 La (re)linéarisation

Dans l'article paru à Crypto'99, Kipnis et Shamir proposent la technique de re-linéarisation.

1. **Paramétriser.** Le système d'équations initial, de m équations linéaires en $\frac{n^2}{2}$ variables y_{ij} et x_i est transformé en une solution paramétrique qui exprime toutes les variables comme une expression linéaire en $M = \frac{n^2}{2} - m = (1-\varepsilon)\frac{n^2}{2}$ nouvelles variables z_i.
2. Ajouter des équations **triviales.** Ajouter **toutes** les équations de degré $c > 1$, qui découlent du fait que $y_{ij} = x_i x_j$. Elles sont appelées des

équations triviales. On les obtient par des permutations de l'ensemble des indices, par example $y_{12}y_{34} = y_{13}y_{24}$ ou $y_{12}y_{37}x_4 = y_{13}y_{24}x_7$.

Pour tout ensemble de K termes qui sont équivalents par permutation d'indices il faut ajouter $K - 1$ équations qui couvrent les K termes. Alors ces équations, malgré le fait qu'elles sont algébriquement dépendantes, seront linéairement indépendantes en tant que polynômes multivariables de degré $\geq c$ en les y_{ij} et x_i.

3. **Substituer** 2. dans 1. Remplacer les y_{ij} et x_i par les z_i :
4. **Résoudre** les équations obtenues par la simple linéarisation si le nombre d'équations dépasse le nombre de termes qui y apparaissent.
5. Sinon on peut **réitérer** 1-3 jusqu'à ce que 4 soit possible.

La technique de relinéarisation telle quelle peut être appliquée sur n'importe quel anneau K.

Dans le cas d'un corps, nous avons montré que :

Résultat [Courtois 1999] L'algorithme de relinéarisation est toujours est 'contenu' dans l'algorithme XL, [42].

La preuve se trouve dans la version étendue de [42] et n'a pas pu être incluse dans le présent mémoire à cause de manque de place.

Par contre, il n'est pas exclu que la technique de relinéarisation soit non-triviale sur un anneau qui n'est pas un corps.

Chapitre 7

L'algorithme XL

Le terme XL veut dire eXtended Linerization, mais aussi multiplier (noté X) et Linéariser. C'est un algorithme particulièrement simple et élégant.

Son intérêt principal, comme on le verra par la suite, réside dans le fait que cet algorithme pourrait résoudre le problème MQ avec une complexité sous-exponentielle. Cela a été pour le première fois suggéré dans notre travail commun avec Patarin, Shamir et Klimov [42].

7.1 Description de l'algorithme XL

Pour les besoins de l'algorithme XL on écrit les équations à résoudre $f(x) = y$ du problème MQ sous la forme particulière suivante :

$$\begin{cases} l_1(x_1, \ldots, x_n) & = & 0 \\ & \vdots & \\ l_m(x_1, \ldots, x_n) & = & 0 \end{cases}$$

Notations : On note l l'ensemble des équations initiales $l = (l_1, \ldots, l_m)$.

Soit $k \in \mathbb{N}$. On dira que les équations de forme $\prod_{j=1}^{k} x_{i_j} * l_i = 0$ sont de type $x^k l$, et on appellera $x^k l$ l'ensemble de toutes ces équations. Par exemple les équations initiales sont de type $x^0 l = l$.

Important : Il faut noter que le degré doit être exactement k et que les degrés sont pris modulo $q - 1$ pour les variables vivant dans $\mathbb{F}_q$.

On note aussi par x^k l'ensemble de tous les termes de degré exactement k, $\prod_{j=1}^{k} x_{i_j}$. C'est une extension (modifiée) de la convention usuelle $x = (x_1, \ldots, x_n)$. On écrit aussi $1 = x^0 = K$, l'ensemble de termes constants.

L'algorithme XL prend un paramètre D, $D \in \mathbb{N}, D \geq 2$.

On considère tous les polynômes $\prod_j x_{i_j} * l_i$ de degré total $\leq D$.

Soit $\mathcal{I}_D$ l'ensemble de ces équations.

$$\mathcal{I}_D = \cup_{k=0}^{D-2} x^k l.$$

On notera $[\mathcal{I}_D]$ l'ensemble des équations qu'elles engendrent (linéairement) $\mathcal{I}_D \subset \mathcal{I}$, où $\mathcal{I}$ est l'idéal engendré par les l_i (On pourrait écrire $\mathcal{I} = [\mathcal{I}_\infty]$).

Définition 7.1.0.1 (L'algorithme XL) *Exécuter les pas suivants :*

1. **Multiplications :** *Générer tous les produits* $\prod_{j=1}^{k} x_{i_j} * l_i \in \mathcal{I}_D$ *avec* $k \leq D-2$.
2. **Linéariser :** *Considérer chaque monôme en les* x_i *de degré* $\leq D$ *comme une nouvelle variable, et faire une élimination Gaussienne de ces variables sur les équations obtenues en 1.*

   L'ordre sur le monômes doit être tel que tous les termes contenant une variable fixée, par exemple x_1*, sont éliminés à la fin.*
3. **Résoudre :** *Supposons que dans l'étape 2 on obtient au moins une équation univariable (avec le puissances de* x_1*). Résoudre cette équation univariable sur un corps fini par les méthodes connues, (par exemple l'algorithme de Berlekamp [3, 65, 70]).*
4. **Répétition :** *Simplifier les équations et répéter l'ensemble de l'attaque afin de trouver la valeur d'autres variables.*

7.1.1 Intérêt de XL

D'une part XL est un algorithme intéressant à étudier parce qu'il est plus simple et meilleur que l'algorithme de relinéarisation de Kipnis [71, 42].

Dans la branche de mathématiques qui étudie la réduction des idéaux de polynômes avec les bases de Gröbner, [40, 43, 42], dont nous parlerons dans 7.6.1, on considère des stratégies d'élimination plus fines que XL et on obtient moins d'équations redondantes (linéairement dépendantes des autres). Il n'est toutefois pas démontré, que la résolution de MQ avec les bases de Gröbner puissent aller considérablement plus vite que le simplissime algorithme XL.

C'est avec XL, et aussi grâce à la relinéarisation, qu'on a pu découvrir (dans l'article d'Eurocrypt'2000 [42]) que :

Le comportement de XL change **radicalement** dès que m dépasse n, voir 7.5.1.

7.2 Example de XL

Soit K un corps et $\mu \neq 0$ dans K. On considère le système d'équations suivant :

$$\begin{cases} x_1^2 + \mu x_1 x_2 = \alpha & (7.2.1) \\ x_2^2 + \nu x_1 x_2 = \beta & (7.2.2) \end{cases}$$

Cela est une instance de MQ avec $m = 2$ and $n = 2$. On remarque que les équations sont homogènes de degré 2. On va considérer XL avec le degré maximal $D = 4$ ce qui correspond à la multiplication des équations par des monômes de degré ≤ 2.

On va faire une restriction supplémentaire par rapport à XL tel que décrit dans le chapitre précédent, on ne va multiplier les équations ci-dessus notées l, seulement par les monômes de degré pair. Les simulations dans 7.4 montreront

que dans le cas des équations homogènes c'est plus intéressant que d'utiliser toutes les monômes, car dans les équations résultantes de XL on aura que de monômes de degré $D, D-2, \ldots$ et l'expérience a montré que cela donne de meilleurs résultats en terme d'indépendance d'équations obtenues.

Ainsi pour $D = 4$ on va utiliser des monômes de degré 2 et 0. Leur ensemble est $x^2 \cup 1$.

Les équations générés dans l'étape 1 de l'algorithm XL sont $l \cup x^2 l \subset \mathcal{I}_4$.

Il s'agit donc de 2 équations initiales et $6 = 2 * 3$ équations supplémentaires générées en multipliant les 2 équations initiales l_i par les 3 termes possibles de degré 2 : $x_1^2, x_1x_2, x_2^2 \in x^2$.

$$\begin{cases} x_1^4 + \mu x_1^3 x_2 = \alpha x_1^2 & (7.2.3) \\ x_1^2 x_2^2 + \nu x_1^3 x_2 = \beta x_1^2 & (7.2.4) \\ x_1^2 x_2^2 + \mu x_1 x_2^3 = \alpha x_2^2 & (7.2.5) \\ x_2^4 + \nu x_1 x_2^3 = \beta x_2^2 & (7.2.6) \\ x_1^3 x_2 + \mu x_1^2 x_2^2 = \alpha x_1 x_2 & (7.2.7) \\ x_1 x_2^3 + \nu x_1^2 x_2^2 = \beta x_1 x_2 & (7.2.8) \end{cases}$$

Dans l'étape 2 de l'algorithme on va éliminer successivement et obtenir :

Avec (7.2.1) : $x_1x_2 = \frac{\alpha}{\mu} - \frac{x_1^2}{\mu}$;

Avec (7.2.2) : $x_2^2 = (\beta - \frac{\alpha\nu}{\mu}) + \frac{\nu}{\mu} x_1^2$;

Avec (7.2.3) : $x_1^3 x_2 = \frac{\alpha}{\mu} x_1^2 - \frac{x_1^4}{\mu}$;

Avec (7.2.4) : $x_1^2 x_2^2 = (\beta - \frac{\alpha\nu}{\mu}) x_1^2 + \frac{\nu}{\mu} x_1^4$;

Avec (7.2.8) : $x_1 x_2^3 = \frac{\alpha\beta}{\mu} + (\frac{\alpha\nu^2}{\mu} - \beta\nu - \frac{\beta}{\mu}) x_1^2 - \frac{\nu^2}{\mu} x_1^4$;

Avec (7.2.6) : $x_2^4 = (\beta^2 - \frac{2\alpha\beta\nu}{\mu}) + (\frac{2\nu\beta}{\mu} + \beta\nu^2 - \frac{\alpha\nu^2}{\mu}) x_1^2 + \frac{\nu^3}{\mu} x_1^4$;

Finalement avec (7.2.5) on obtient une équation de degré $D = 4$ en **une seule** variable x_1 :

$$\alpha^2 + x_1^2(\alpha\mu\nu - \beta\mu^2 - 2\alpha) + x_1^4(1 - \mu\nu) = 0.$$

C'est le but recherché par l'algorithme XL.

7.3 Analyse asymptotique de faisabilité de XL

Le nombre de termes de degré $\leq D$ avec n variables sur un corps de grand cardinal est égal au nombre de termes de degré exactement D avec $n+1$ variables qui est :

$$\leq \binom{n+1+D-1}{D-1}.$$

On suppose $D << n \leq m$. Le nombre d'équations dans $\mathcal{I}_D$ est estimé à :

$$All \approx m \cdot \binom{n+D-2}{D-3} \approx m \cdot n^{D-3}/(D-3)!$$

On supposera dans cette analyse simplifiée que la plupart d'entre elles sont linéairement indépendantes,

$$Free \approx All.$$

Le nombre de termes T dans ces équations est de l'ordre du cardinal de x^D :

$$T \approx n^D/D!$$

L'algorithme XL marche dès que $Free \approx T$.

$$n^D/D! \approx mn^{D-2}/(D-2)!$$
$$n^2 \approx mD(D-1)$$
$$\boxed{D \approx \frac{n}{\sqrt{m}}}$$

7.4 Des simulations sur XL

Le but de ces simulations est de voir pour quel D minimal, l'algorithme XL est susceptible de fonctionner.

Nous donnerons ici bien davantage de simulations que dans l'article [42].

Les tableaux décrivent :

1. Les types d'équations considérés, par exemple $xl \cup l$
2. Le nombre de ces équations (**All**) et le nombre de celles qui sont linéairement indépendantes (**Free**).
3. Les types de termes qui sont présents dans ces équations par exemple $x^3 \cup x^2$
4. Le nombre de ces termes **T**.
5. Le nombre **B** de termes en une seule variable (p.ex.x_1).

Avec ces notations, il devient possible d'éliminer tous les T termes les B termes en une seule variable dès que :

$$\Delta \geq 0$$

avec

$$\Delta = Free + B - T - 1.$$

7.4.1 Le plus petit D pour $m = n, n+1, n+2, \ldots$

Les simulations avec $m = n$

2 variables and 2 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	2/2	-1	1	3	x^2
$xl \cup l$	6/6	-1	3	9	$x^3 \cup x^2 \cup x$
x^2l	6/6	-1	2	8	$x^4 \cup x^2$
$x^2l \cup l$	7/8	0	2	8	$x^4 \cup x^2$
$x^2l \cup xl \cup l$	11/12	0	4	14	$x^4 \cup x^3 \cup x^2 \cup x$
x^3l	8/8	-1	2	10	$x^5 \cup x^3$
$x^3l \cup xl$	10/12	1	3	11	$x^5 \cup x^3 \cup x$
$x^3l \cup xl \cup l$	12 /14	1	4	14	$x^5 \cup x^3 \cup x^2 \cup x$
$x^3l \cup x^2l \cup xl \cup l$	17/20	1	5	20	$x^5 \cup \ldots$
x^4l	10/10	-1	2	12	$x^6 \cup x^4$
$x^4l \cup x^2l \cup l$	14/18	1	3	15	$x^6 \cup x^4 \cup x^2$
$x^4l \cup x^3l \cup x^2l \cup xl \cup l$	24/30	2	6	27	$x^6 \cup \ldots$

2 variables and 2 non-homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	2/2	-2	2	5	$x^2 \cup x$
$xl \cup l$	6/6	-1	3	9	$x^3 \cup x^2 \cup x$
x^2l	6/6	-5	4	14	$x^4 \cup x^3 \cup x^2 \cup x$
$x^2l \cup xl \cup l$	11/12	0	4	14	$x^4 \cup x^3 \cup x^2 \cup x$
x^3l	8/8	-5	3	15	$x^5 \cup x^4 \cup x^3$
$x^3l \cup x^2l \cup xl \cup l$	16/20	0	5	20	$x^5 \cup \ldots$
x^4l	10/10	-5	4	18	$x^6 \cup x^5 \cup x^4$
$x^4l \cup x^3l \cup x^2l \cup xl \cup l$	24/30	2	6	27	$x^6 \cup \ldots$

Δ ≥ 0 when XL solves the equations, Δ = Free+B-T-1 **T**: number of monomials

Free/All: nb. of free/all equations of given **type** **B**: nb. of monomials in one variable e.g. x_1

3 variables and 3 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	3/3	-3	1	6	x^2
$xl \cup l$	12/12	-5	3	19	$x^3 \cup x^2 \cup x$
x^2l	17/18	-3	2	21	$x^4 \cup x^2$
$x^2l \cup l$	18/21	-2	2	21	$x^4 \cup x^2$
$x^2l \cup xl \cup l$	27/30	-4	4	34	$x^4 \cup x^3 \cup x^2 \cup x$
x^3l	27/30	-3	2	31	$x^5 \cup x^3$
$x^3l \cup xl$	30/39	-2	3	34	$x^5 \cup x^3 \cup x$
$x^3l \cup xl \cup l$	33/42	-3	5	40	$x^5 \cup x^3 \cup x^2 \cup x$
$x^3l \cup x^2l \cup xl \cup l$	48/84	-3	5	55	$x^5 \cup \dots$
x^4l	39/45	-3	2	43	$x^6 \cup x^4$
$x^4l \cup x^2l$	45/63	-2	3	49	$x^6 \cup x^4 \cup x^2$
$x^4l \cup x^2l \cup l$	46/66	-1	3	49	$x^6 \cup x^4 \cup x^2$
$x^4l \cup x^3l \cup \dots$	76/105	-2	6	83	$x^6 \cup \dots$
x^5l	53/63	-2	2	57	$x^7 \cup x^5$
$x^5l \cup x^3l$	63/93	-2	3	67	$x^7 \cup x^5 \cup x^3$
$x^5l \cup x^3l \cup xl$	66/102	-1	4	70	$x^7 \cup x^5 \cup x^3 \cup x$
$x^5l \cup x^4l \cup \dots$	112/168	-1	7	119	$x^7 \cup \dots$
x^6l	69/84	-3	2	73	$x^8 \cup x^6$
$x^6l \cup x^4l$	84/129	-2	3	88	$x^8 \cup x^6 \cup x^4$
$x^6l \cup x^4l \cup x^2l$	90/147	-1	4	94	$x^8 \cup x^6 \cup x^4 \cup x^2$
$x^6l \cup x^4l \cup x^2l \cup l$	91/150	0	4	94	$x^8 \cup x^6 \cup x^4 \cup x^2$

3 variables and 3 non-homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	3/3	-5	2	9	$x^2 \cup x$
$xl \cup l$	12/12	-5	3	19	$x^3 \cup x^2 \cup x$
$x^2l \cup xl \cup l$	27/30	-4	4	34	$x^4 \cup x^3 \cup x^2 \cup x$
$x^3l \cup x^2l \cup xl \cup l$	48/84	-3	5	55	$x^5 \cup \dots$
$x^4l \cup x^3l \cup \dots$	76/	-2	6	83	$x^6 \cup \dots$
$x^5l \cup x^4l \cup \dots$	112/328	-1	7	119	$x^7 \cup \dots$
$x^6l \cup x^4l^2 \cup \dots$	157/577	0	8	164	$x^8 \cup \dots$

Δ ≥ 0 when XL solves the equations, $\Delta =$ Free+B-T-1 **T**: number of monomials

Free/All: nb. of free/all equations of given **type** **B**: nb. of monomials in one variable e.g. x_1

4 variables and 4 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	4/4	-6	1	10	x^2
$xl \cup l$	20/20	-12	3	34	$x^3 \cup x^2 \cup x$
$x^2l \cup l$	38/44	-6	2	45	$x^4 \cup x^2$
$x^2l \cup xl \cup l$	54/60	-11	4	69	$x^4 \cup x^3 \cup x^2 \cup x$
$x^3l \cup xl$	72/96	-6	3	80	$x^5 \cup x^3 \cup x$
$x^3l \cup x^2l \cup xl \cup l$	111/140	-10	5	125	$x^5 \cup \ldots$
x^4l	111/140	-7	2	119	$x^6 \cup x^4$
$x^4l \cup x^2l$	121/180	-7	3	129	$x^6 \cup x^4 \cup x^2$
$x^4l \cup x^2l \cup l$	122/184	-5	3	129	$x^6 \cup x^4 \cup x^2$
$x^4l \cup x^3l \cup \ldots \cup l$	194/280	-10	6	209	$x^6 \cup \ldots$
x^5l	168/224	-7	2	176	$x^7 \cup x^5$
$x^5l \cup x^3l$	188/304	-6	3	196	$x^7 \cup x^5 \cup x^3$
$x^5l \cup x^3l \cup xl$	192/320	-7	4	200	$x^7 \cup x^5 \cup x^3 \cup x$
$x^5l \cup x^4l \cup \ldots$	314/504	-9	7	329	$x^7 \cup \ldots$
x^6l	241/336	-7	2	249	$x^8 \cup x^6$
$x^6l \cup x^4l$	276/476	-6	3	284	$x^8 \cup x^6 \cup x^4$
$x^6l \cup x^4l \cup x^2l \cup l$	287/520	-4	4	294	$x^8 \cup x^6 \cup x^4 \cup x^2$
$x^6l \cup x^5l \cup \ldots$	479/840	-8	8	494	$x^8 \cup \ldots$
x^7l	332/480	-7	2	340	$x^9 \cup x^7$
$x^7l \cup x^5l$	388/704	-6	3	396	$x^9 \cup x^7 \cup x^5$
$x^7l \cup x^5l \cup x^3l$	408/784	-5	4	416	$x^9 \cup x^7 \cup x^5 \cup x^3$
$x^7l \cup x^5l \cup x^3l \cup xl$	412/800	-4	5	420	$x^9 \cup x^7 \cup \ldots$
$x^7l \cup x^6l \cup \ldots$	699/	-7	9	714	$x^9 \cup \ldots$
x^8l	443/660	-7	2	451	$x^{10} \cup x^8$
$x^8l \cup x^6l \cup \ldots$	573/1180	-3	5	580	$x^{10} \cup x^8 \cup \ldots$
$x^8l \cup x^7l \cup \ldots$	985/1980	-6	10	1000	$x^{10} \cup \ldots$
$x^9l \cup x^7l \cup \ldots$	776/1680	-3	6	784	$x^{11} \cup x^9 \cup \ldots$
$x^9l \cup x^8l \cup \ldots$	1349/2860	-5	11	1364	$x^{11} \cup \ldots$
$x^{10}l \cup x^8l \cup \ldots$	1028/2324	-2	6	1035	$x^{12} \cup x^{10} \cup \ldots$
$x^{10}l \cup x^9l \cup \ldots$	1804/4004	-5	12	1819	$x^{12} \cup \ldots$
$x^{11}l \cup x^9l \cup \ldots$	1336/3136	-2	7	1344	$x^{13} \cup x^{11} \cup \ldots$
$x^{11}l \cup x^{10}l \cup \ldots$	2364/5460	-3	13	2379	$x^{13} \cup \ldots$
$x^{12}l \cup x^{10}l \cup \ldots$	1708/4144	-1	7	1715	$x^{14} \cup x^{12} \cup \ldots$
$x^{12}l \cup x^{11}l \cup \ldots$	3044/7280	-2	14	3059	$x^{14} \cup \ldots$
$x^{13}l \cup x^{11}l \cup \ldots$	2152/5376	-1	8	2160	$x^{15} \cup \ldots$
$x^{13}l \cup x^{12}l \cup \ldots$	3860/9520	-1	15	3875	$x^{15} \cup \ldots$
$x^{14}l \cup x^{12}l \cup \ldots$	2677/6864	0	8	2684	$x^{16} \cup x^{14} \cup \ldots$

Interprétation

Dans les tableaux XL fonctionne quand $\Delta \geq 0$. On voit que XL avec $m = n$ fonctionne (seulement) pour $D \geq 2^n$.

Les simulations avec $m = n + 1$

4 variables and 5 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	5/5	-4	1	10	x^2
$xl \cup l$	25/25	-8	3	34	$x^3 \cup x^2 \cup x$
$x^2l \cup l$	45/55	1	2	45	$x^4 \cup x^2$
$x^2l \cup xl \cup l$	65/75	-1	4	69	$x^4 \cup x^3 \cup x^2 \cup x$

5 variables and 6 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	6/6	-9	1	15	x^2
xl	30/30	-9	2	40	$x^3 \cup x$
$xl \cup l$	36/36	-17	3	55	$x^3 \cup x^2 \cup x$
$x^2l \cup l$	81/96	-3	2	85	$x^4 \cup x^2$
$x^2l \cup xl \cup l$	111/126	-11	4	125	$x^4 \cup x^3 \cup x^2 \cup x$
$x^3l \cup xl$	165/240	2	3	166	$x^5 \cup x^3 \cup x$

6 variables and 7 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	7/7	-14	1	21	x^2
xl	42/42	-19	2	62	$x^3 \cup x$
$x^2l \cup l$	133/154	-13	2	147	$x^4 \cup x^2$
$x^3l \cup xl$	308/434	-4	3	314	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	609/1036	2	3	609	$x^6 \cup x^4 \cup x^2$

7 variables and 8 homogenous quadratic equations, $GF(127)$

XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	8/8	-20	1	28	x^2
xl	56/56	-34	2	91	$x^3 \cup x$
$x^2l \cup l$	204/232	-33	2	238	$x^4 \cup x^2$
$x^3l \cup xl$	532/728	-19	3	553	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	1156/1912	-4	3	1162	$x^6 \cup x^4 \cup x^2$
$x^5l \cup x^3l \cup xl$	2268/4424	2	4	2269	$x^7 \cup x^5 \cup x^3 \cup x$

8 variables and 9 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	9/9	-27	1	36	x^2
xl	72/72	-53	2	128	$x^3 \cup x$
$x^2l \cup l$	297/333	-68	2	366	$x^4 \cup x^2$
$x^3l \cup xl$	864/1152	-54	3	920	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	2055/3303	-25	3	2082	$x^6 \cup x^4 \cup x^2$
$x^5l \cup x^3l \cup xl$	4344/8280	-5	4	4352	$x^7 \cup x^5 \cup x^3 \cup x$
$x^6l \cup x^4l \cup x^2l \cup l$	8517/18747	3	4	8517	$x^8 \cup x^6 \cup x^4 \cup x^2$

Interprétation On remarque que XL avec $m = n + 1$ fonctionne quand $D \geq n$.

Les simulations avec $m = n + 2$

8 variables and 10 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	free/all	(free+B-T-1)		T	type
l	10/10	-26	1	36	x^2
xl	80/80	-47	2	128	$x^3 \cup x$
$x^2l \cup l$	325/370	-40	2	366	$x^4 \cup x^2$
$x^3l \cup xl$	919/1280	1	3	920	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	2082/3670	2	3	2082	$x^6 \cup x^4 \cup x^2$

9 variables and 11 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	11/11	-34	1	45	x^2
$x^3l \cup xl$	1419/1914	-40	3	1461	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	3543/5951	2	3	3543	$x^6 \cup x^4 \cup x^2$

10 variables and 12 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	12/12		1	55	x^2
$x^4l \cup x^2l \cup l$	5775/9252	2	3	5775	$x^6 \cup x^4 \cup x^2$

Interprétation On remarque que quand $m = n + 2$ la valeur de D pour laquelle $\Delta \geq 0$ et l'algorithme fonctionne s'effondre brusquement par rapport aux valeurs précédentes de 2^n et n.

Les simulations avec $m = n + 3$

8 variables and 11 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	11/11	-26	1	36	x^2
xl	88/88	-39	2	128	$x^3 \cup x$
$x^2l \cup l$	352/407	-13	2	366	$x^4 \cup x^2$
$x^3l \cup xl$	919/1408	1	3	920	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	2082/4037	2	3	2082	$x^6 \cup x^4 \cup x^2$

12 variables and 15 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	15/15	-63	1	78	x^2
$x^4l \cup x^2l \cup l$	13819/21660	2	3	13819	$x^6 \cup x^4 \cup x^2$

Interprétation On remarque que quand $m = n + 3$ la valeur de D s'effondre brusquement et pourrait tendre vers la valeur conjecturée $D = \mathcal{O}(\sqrt{n})$.

Les simulations avec $m = n + 4$

8 variables and 12 homogenous quadratic equations, $GF(127)$					
XL equations		Δ	B	XL unknowns (B degrees)	
type	Free/All	(Free+B-T-1)		T	type
l	12/12	-26	1	36	x^2
xl	96/96	-31	2	128	$x^3 \cup x$
$x^2l \cup l$	366/444	1	2	366	$x^4 \cup x^2$
$x^3l \cup xl$	919/1536	1	3	920	$x^5 \cup x^3 \cup x$
$x^4l \cup x^2l \cup l$	2082/4404	2	3	2082	$x^6 \cup x^4 \cup x^2$

Interprétation On remarque que quand $m = n + 4$ la valeur de D s'effondre brusquement et semble tendre la conjecturée $D = \mathcal{O}(\sqrt{n})$.

7.4.2 Le plus petit m pour n, D fixés.

Les simulations avec le meilleur m pour $n = 8$

8 variables, homogenous quadratic equations, $GF(127)$				
		Δ	B	
l equations				x^2 monomials
initial m	Free/All			number (T)
36	36/36	0	1	36
xl equations				$x^3 \cup x$ monomials
initial m	Free/All			number (T)
16	127/128	0	2	128
$x^2 l \cup l$ equations				$x^4 \cup x^2$ monomials
initial m	Free/All			number (T)
10	325/370	-40	2	366
11	352/407	-13	2	366
12	366/444	1	2	366
$x^3 l \cup xl$ equations				$x^5 \cup x^3$ monomials
initial m	Free/All			number (T)
10	919/1280	1	3	920
$x^4 l \cup x^2 l \cup l$ equations				$x^6 \cup x^4 \cup x^2$ monomials
initial m	Free/All			number (T)
9	2055/3303	-25	3	2082
10	2082/3670	2	3	2082
$x^5 l \cup x^3 l \cup xl$ equations				$x^7 \cup x^5 \cup x^3 \cup x$ monomials
initial m	Free/All			number (T)
9	4344/8280	-5	4	4352
10	4351/9200	2	4	4352
$x^6 l \cup x^4 l \cup x^2 l$ equations				$x^8 \cup x^6 \cup x^4 \cup x^2$ monomials
initial m	Free/All			number (T)
9	8517/18747	3	4	8517

Δ ≥ 0 when XL solves the equations, Δ = Free+B-T-1 — **T**: number of monomials

Free/All: nb. of free/all equations of given **type** — **B**: nb. of monomials in one variable e.g. x_1

Les simulations avec le meilleur m pour $n = 15$

15 variables, homogenous quadratic equations, $GF(127)$

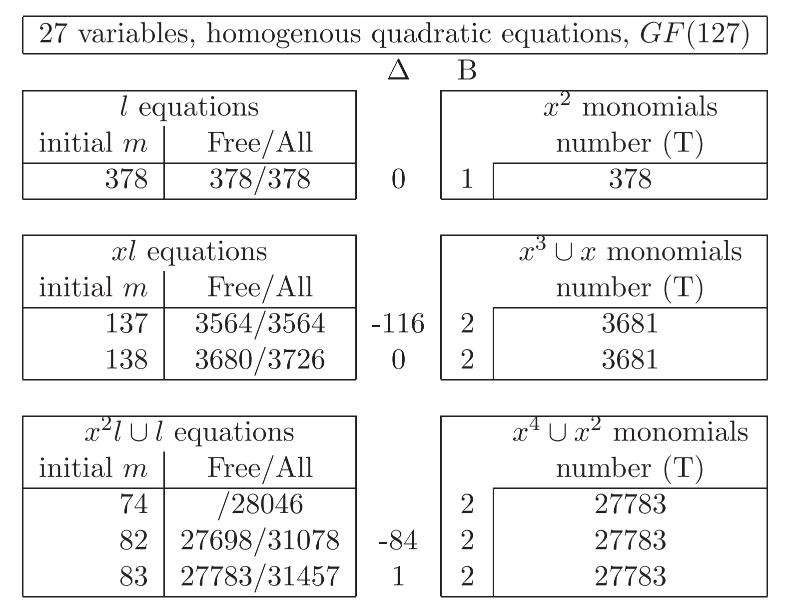
Equations	initial m	Free/All	Δ	B	Monomials	number (T)
l equations	120	120/120	0	1	x^2 monomials	120
xl equations	46	690/690	-4	2	$x^3 \cup x$ monomials	695
	47	694/705	0	2		695
$x^2l \cup l$ equations	27	2916/3267	-263	2	$x^4 \cup x^2$ monomials	3180
	29	3180/3509	1	2		3180
$x^3l \cup xl$ equations	19	10640/13205	-1683	3	$x^5 \cup x^3 \cup x$ monomials	12323
	24	12322/16680	1	3		12323

Les simulations avec le meilleur m pour $n = 18$

18 variables, homogenous quadratic equations, $GF(127)$

Equations	initial m	Free/All	Δ	B	Monomials	number (T)
l equations	171	171/171	0	1	x^2 monomials	171
xl equations	65	1157/1170	0	2	$x^3 \cup x$ monomials	1158
$x^2l \cup l$ equations	36	5562/6192	-594	2	$x^4 \cup x^2$ monomials	6156
	41	6156/7052	1	2		6156
$x^3l \cup xl$ equations	24	22824/27792		3	$x^5 \cup x^3 \cup x$ monomials	27492
	31	27491/35898	1	3		27492

Les simulations avec le meilleur m pour $n = 27$



27 variables, homogenous quadratic equations, $GF(127)$

equations	initial m	Free/All	Δ	B	monomials	number (T)
l equations	378	378/378	0	1	x^2 monomials	378
xl equations	137	3564/3564	-116	2	$x^3 \cup x$ monomials	3681
	138	3680/3726	0	2		3681
$x^2l \cup l$ equations	74	/28046		2	$x^4 \cup x^2$ monomials	27783
	82	27698/31078	-84	2		27783
	83	27783/31457	1	2		27783

Δ ≥ 0 when XL solves the equations, Δ = Free+B-T-1 **T**: number of monomials

Free/All: nb. of free/all equations of given **type** **B**: nb. of monomials in one variable e.g. x_1

Interprétation des résultats

On sait que le nombre m d'équations nécessaires pour résoudre par XL vaut $n^2/2$ avec $D = 2$.

On observe qu'il s'effondre rapidement quand D croît, et tend vers n.

7.5 Analyse de XL

7.5.1 Interprétation des simulations

Quand $m \approx n$, on a vu dans 7.3 que l'on s'attend à avoir

$$D \approx \frac{n}{\sqrt{m}} \approx \sqrt{n}.$$

Les simulations ont montré que D vaut :

- $D = 2^n$ quand $m = n$.
- $D = n$ quand $m = n + 1$.
- D décroît très vite pour $m = n + 2$.

  ⋮
- Il semble plausible que $D \to \mathcal{O}(\sqrt{n})$.

Notre estimation asymptotique $D \approx \frac{n}{\sqrt{m}}$ semble être vraie quand m dépasse de peu n.

D'avantage de simulations sont nécessaires pour avoir des estimations plus précises.

7.5.2 Analyse de complexité de XL

Sur cette base on estime la complexité de XL, qui est la complexité de l'élimination Gaussienne finale de XL.

La taille d'équations (nombre de termes) est environ :

$$T = n^D/D!$$

Soit ω tel que la complexité de la réduction Gaussienne d'un système $n \times n$ soit n^ω. On a

$$2 \leq \omega < 3$$

La complexité de XL est :

$$WF \;=\; T^\omega \;\approx\; n^{\omega D}/D!$$

(WF=Working Factor, le facteur de travail).

On ne peut pas en dire plus sur la valeur réaliste de ω à prendre.

D'une part, le meilleur exposant théorique connu est $\omega = 2.376$ [90]. En pratique il peut s'avérer que la constante dans les meilleurs algorithmes asymptotiques soit trop grande. D'après [90] personne n'a encore estimé cette constante. Il peut donc s'avérer que $\omega = 3$ en pratique.

D'autre part les équations de XL peuvent être considérées comme 'sparses'. Elles ne sont toutefois pas assez 'sparses' pour affirmer à coup sûr que $\omega \approx 2$ ce qui est obtenu pour les systèmes d'équations très sparses.

7.5.3 Complexité quand $m = \varepsilon n^2$

Pour un système de $m = \varepsilon n^2$ équations avec n variables on obtient :

$$D \approx \frac{n}{\sqrt{m}} \approx \lceil \frac{1}{\sqrt{\varepsilon}} \rceil$$

Donc XL devrait résoudre un système de $m = \varepsilon n^2$ équations avec n variables en temps **polynomial** de l'ordre de :

$$WF \;=\; T^\omega \;\approx\; n^{\omega D}/D! \;\approx\; \mathcal{O}\left(n^{\frac{\omega}{\sqrt{\varepsilon}}}\right).$$

7.5.4 Cas $m \approx n$, FXL

Pour résoudre le cas $m \approx n$ nous utilisons une variante de l'algorithme XL.

Comme on l'a vu, le comportement de XL change de façon très brutale quand m dépasse n. Ainsi il est très intéressant de pouvoir donc diminuer (de peu) n. Puisqu'on est sur un corps fini, cela est possible en devinant la valeur de quelques variables. Ce n'est qu'ensuite que l'on appliquera XL.

Combien de variables doivent être devinées ? Cela n'est pas très clair.

Supposons que $m = n$ et que l'on devine environ $\sqrt{n}$ variables.

7.5.5 Complexité de FXL quand $m = n$

Alors FXL devrait résoudre un système de n équations quadratiques avec n variables sur un corps fini $\mathbb{F}_q$ en un temps **sous-exponentiel** :

$$WF \approx q^{\sqrt{n}} n^{\omega\sqrt{n}}$$

7.6 L'apport de la méthode XL

Le fait que le problème MQ soit polynomial quand $m = \varepsilon n^2$ est naturel pour la méthode de relinéarisation. Shamir et Kipnis ont été les premiers à le suggérer dans [71].

Or nous avons montré que la relinéarisation est contenue dans XL [42]. XL est donc probablement polynomial en moyenne pour $m = \varepsilon n^2$.

De plus, quand $m = n$, il semble que le problème MQ pourrait même être résolu en temps sous-exponentiel par FXL. De futures simulations apporteront davantage des précisions à ce sujet.

Ces résultats ont apparemment surpris de nombreuses personnes qui ont depuis une vingtaine d'années étudié le domaine de mathématiques appelé les bases de Gröbner.

7.6.1 XL et les bases de Gröbner

En principe les bases de Gröbner, [40, 43, 42], donnent des stratégies plus fines que XL pour obtenir le même résultat tout en générant moins d'équations que dans XL.

Notons toutefois que d'après Jean-Charles Faugère il n'est pas certain que les bases de Gröbner puissent en pratique résoudre plus de choses que XL.

7.6.2 MQ en cryptographie et MQ en bases de Gröbner

En cryptographie nous voulons d'habitude résoudre un MQ comme suit :

- K est un petit corps fini $K = \mathbb{F}_q$, par exemple $\mathbb{F}_{16}$.
- La caractéristique p est petite, par exemple 2.
- Souvent le système est suffisamment défini ou sur-défini, $m \geq n$.
- Le système admet une solution, ce qui n'est pas naturel pour un système aléatoire avec $m > n$.

Il semble que, dans la vaste littérature existante sur les bases de Gröbner on a d'habitude :

- K est algébriquement clos, donc infini.
- La caractéristique p est 0.
- Le système est exactement défini $m = n$.
- Le système est totalement aléatoire.

Ainsi, on est passé à côté de notre découverte d'algorithme polynomial XL pour MQ avec $m = \varepsilon n^2$[42] et de l'algorithme FXL qui est probablement sous-exponentiel même pour $m = n$. .

7.7 Conclusion, XL en pratique

En pratique, pour les valeurs utilisées pour les cryptosystèmes comme HFE, XL s'avère beaucoup moins rapide que la recherche exhaustive.

Nous avons estimé que le seuil où XL serait plus rapide que la recherche exhaustive pour résoudre n équations avec n inconnues sur $\mathbb{F}_2$, serait de l'ordre de :

$$n > 100$$

Et toujours, pour de tels n, la complexité serait beaucoup trop grande pour les ordinateurs actuels qui peuvent faire jusqu'à environ 2^{50} calculs par seconde.

Il semble donc raisonnable de faire de la cryptographie avec des polynômes multivariables.

7.8 Appendice - XL et équations cubiques

2 variables, non-homogenous cubic $GF(127)$

Equations	initial m	Free/All	Δ	B	Monomials	number (T)
l equations	5	9/9	0	3	$x^3 \cup x^2 \cup x$ monomials	9
$xl \cup l$ equations	4	12/12	1	4	$x^4 \cup x^3 \cup \ldots$ monomials	14
$x^2l \cup xl \cup l$ equations	3	18/18	2	5	$x^5 \cup ldots$ monomials	20
$x^6l \cup \ldots$ equations	2	46/56	0	9	$x^9 \cup \ldots$ monomials	54

3 variables, non-homogenous cubic $GF(127)$

Equations	initial m	Free/All	Δ	B	Monomials	number (T)
l equations	17	17/17	0	3	$x^3 \cup x^2 \cup x$ monomials	19
$xl \cup l$ equations	8	32/32	1	4	$x^4 \cup x^3 \cup \ldots$ monomials	34
$x^2l \cup xl \cup l$ equations	6	55/60	4	5	$x^5 \cup ldots$ monomials	55
$x^3l \cup x^2l \cup \ldots$ equations	5	83/100	0	6	$x^6 \cup x^5 \cup \ldots$ monomials	83
$x^4l \cup x^3l \cup \ldots$ equations	4	116/140	3	7	$x^7 \cup x^6 \cup \ldots$ monomials	119
$x^{24}l \cup \ldots$ equations	3	/	0	27	$x^{27} \cup \ldots$ monomials	4059

Chapitre 8

Algorithmes pour MQ avec $m << n$

Cette partie est rédigée en anglais.

Le but de ces algorithmes est de casser MQ quand $m << n$. Ils sont également intéressants pour $m = n$ avec q^n proche de le limite la puissance de calcul disponible (algorithmes quasi-exhaustifs). Par exemple quand $q^n = 2^{64}$ il est tentant de voir si on ne peut avoir une attaque en 2^{60}. Il faut toutefois se rappeler que les complexités données sont des approximations, et un triplet q, m, n concret demande une évaluation plus précise.

La complexité de (nombreuses) attaques de cette partie est de l'ordre de

$$WF = q^{Max(m/2, m-\sqrt{n/2})}$$

ou moins, mais alors il y a le risque que le facteur polynomial dans la complexité devienne trop grand.

On remarque que la limite quand $n \to \infty$ avec m fixé est de

$$WF \to q^{m/2}$$

Pour des corps de caractéristique 2, dans l'article [76], on trouve un autre algorithme qui est polynomial dès que $n \geq m^2$. Le problème d'obtenir le même résultat sur un corps de caractéristique > 2 reste ouvert.

De même, toujours pour des corps de caractéristique 2, dans un travail récent qui n'est pas encore terminé [45], les auteurs proposent un algorithme très différent des nôtres, dont la complexité semble être d'environ $q^{m-log_2(n/m)}$.

8.1 Useful problems related to MQ

Let f be a quadratic multivariate function $f : GF(q)^n \to GF(q)^m$.

We suppose that f has no constant terms, and the equation to solve is $f(x) = c$ with a constant $c \in GF(q)^m$. We usually suppose $n \geq m$ which implies that there are q^{n-m} solutions in average.

The following problems are closely related :

1. **One-Solving :** Find one x such that $f(x) = c$ in less than $\mathcal{O}(q^m)$.

2. **Multi-Solving :** Find M solutions x such that $f(x) = c$ in less than $\mathcal{O}(Mq^m)$. We present below several versions of this problem.
3. **All-Solving :** Find all x such that $f(x) = c$ in less than $\mathcal{O}(q^n)$.
4. **Solving-in-the-Middle :** Find M solutions to $f(x) = g(x) + c$ in less than $\mathcal{O}(\sqrt{M} * q^{m/2})$.

The main focus is the first problem, however as we will see, in our best algorithm for One-Solving we used both an algorithm for 4. and for a special case of 3.

8.2 Solving-in-the-Middle.

The problem [Solving-in-the-Middle] : Find M solutions to $f(x) = f'(x') + c$.

We present the standard birthday-paradox algorithm in $\mathcal{O}(\sqrt{M} * q^{m/2})$.

Working condition : $n \geq m/2 + \frac{\log M}{2 \log q}$, it also assures that such a system is expected to have indeed M solutions.

Solution : We generate $\sqrt{M} * q^{m/2}$ random x and $\sqrt{M} * q^{m/2}$ random x'. The number of possible differences $f(x) - f'(x')$ is about Mq^m, therefore the value c is reached about M times.

In the following chapters we will use this result applied to a part of equations $f_{1..k}$.

Small improvement. For quadratic equations we can decrease the working time of the algorithm by 2, provided that n exceeds $m/2 + \frac{\log M}{2 \log q}$ by a constant value, using differential solving on f and f' and working systematically with pairs of x and $x + z$ such that $f(x)$ and $f(x + z)$ are the same. We have then 4 solutions for every collision $f(x) - f'(x') = c$.

8.3 Multi-solving.

At present it appears to be an open problem (in general) and there are several possible goals to achieve :

1. The goal is to compute M solutions to $f(x) = c$ faster than the exhaustive search in Mq^m.
2. The goal is to compute M solutions to a system not only faster than the exhaustive search, but also faster that the time to find M times one solution by our best algorithm.
3. The goal is 1. or 2. but we require that the set of solutions is a linear space.

8.4 Special case Multi-solving.

The special case multi-solving we are interested in, are those subcases that can be generated out of a general random system, by the two following methods :
- Eliminating terms via linear variable changes on the outputs (equations).
- Eliminating terms via linear variable changes on the inputs (variables).

So far, in all our attacks, we have only make use of the first possibility. Four cases are studied below.

Conventions for Multi-Solving

After the variable changes, we note :
- **k** variables out of n are called $x_1, \ldots, x_k$
- the remaining $\mathbf{n} - k$ are called $x'_{k+1}, \ldots, x'_n$.

We suppose that we managed to get a subsystem of k' equations out of m with one of the following properties :

1. The terms in $x_i x'_j$, do not appear in $f_1, \ldots, f_{k'}$.
2. The terms in $x_i x_j$, do not appear in $f_1, \ldots, f_{k'}$.
3. Both the above do not appear, only linear terms in x_i.
4. The variables x_i are wholly absent in $f_1, \ldots, f_{k'}$.

Below we present algorithms for Multi-Solving in each of these cases.

Special case Multi-solving 1.

Without the terms of type xx', a function f can be split in 2 independent parts, and we are exactly in the case of Solving-in-the-Middle described in 8.1.

Special case Multi-solving 2.

Without the terms of type $x_i x_j$ the equations become linear in x_i once x' is fixed. Therefore :
- If $k \geq m$, for an arbitrarily chosen x' we get about q^{m-k} solutions (x, x').
- Conversely if $k < m$ we get one solution with a probability about q^{m-k}.

Special case Multi-solving 3.

In this case, the equations are linear in x_i before x' is chosen.
Therefore :
- If $k \geq m$, we get q^{m-k} solutions (x, x'), with the $(x_i - c_i)$ being expressed directly as a parameterized linear space and a linear function of quadratic equations in the x'_i.
- If $k < m$ for a random x' we get a solution x only with a probability about q^{m-k}. However we can eliminate the x_i and derive $m - k$ quadratic equations on the x'_i only. Then make use of special case Multi-solving 4 for those $m - k$ equations. Every solution found for those $m - k$ equations is expected to extend to all the m equations. (We precompute an expression of each x_i directly as a set of quadratic functions in x'_j.)

Special case Multi-solving 4.

Each solution x' found gives a linear space of solutions : (x, x') with any x.

We get Mq^k solutions. And we need M times the time to find a solution to a system of m equations and $n-k$ variables, which is at most $\mathcal{O}(Mq^m)$ computations.

8.5 Advanced algorithms for MQ $m << n$.

The attacks we present are called D, E, FL, FXL, EFL and EFXL.

The best attack is EFXL however the polynomial part of the complexity of EFXL is larger than in EFL. In practice we expect EFL to be the best.

Many of the attacks we present are based on the principle of Differential Solving :

$$f_l(x+\alpha) - f_l(x) - f_l(\alpha) \quad = \quad \sum_{ij} p_{lij} x_i \alpha_j.$$

Usually we construct 2 linear subspaces X and A of $GF(q)^n$ such that if $(x, \alpha) \in X \times A$, then $f(x+\alpha) - f(x)$ has some special properties (that vary from one attack to another).

8.6 The 'D' attack in $q^{m-\log_q(n-m)}$.

Let $g : GF(q)^k \to GF(q)^n$ and $h : GF(q)^k \to GF(q)^m$ be two **random** functions.

In this attack we pick the first k equations out of m and we have

$$f_l(x+\alpha) - f_l(x) \quad = \quad f_l(\alpha) + \sum_{ij} p_{lij} x_i \alpha_j \qquad \forall_{l=1..k}$$

For a fixed α, $f(x+\alpha) - f(x)$ expresses as k linear equations in n variables. We want to force these k equations to be true. We consider the set X of values x that satisfy :

$$X = \{\ x \ \mid\ \forall_{b \in GF(q)^k} \ \ \forall_{l=1..k} \ \ f_l(x+g(b)) - f_l(x) = h(b)\}$$

First we note that X ia a linear space. For each b we have imposed k linear constraints on the x_i. In all if we assume that :

$$n - k(q^k) \geq m - k$$

then, the dimension of X is still at least $m-k$.

The attack. Given a fixed c, we are going to solve the equation $f(x + alpha) = c$.

1. Let $k \approx \log_q(n-m)$ (as we need to have $q^{n-k(q^k)} \geq q^m$).
2. Let $h : (b_1, \ldots, b_k) \mapsto (b_1, \ldots, b_k, 0, \ldots, 0)$ and g be any injective function.
3. We pick a random $x \in X$ out of at least q^{m-k}.

4. Now for all b

$$f_{1..k}(x + g(b)) - f_{1..k}(x) \ = \ h_{1..k}(b) = b,$$

therefore if $b = c_{1..k} + f_{1..k}(x)$

$$f_{1..k}(x - g(c_{1..k} + f_{1..k}(x))\ ) = c_{1..k}.$$

5. With probability q^{k-m}, the chosen x also satisfies the remaining $m - k$ equations

$$f_{k+1...m}(x - g(c_{1..k} - f_{1..k}(x))\ ) \stackrel{?}{=} c_{k+1...m}.$$

6. We repeat the attack about q^{m-k} times.

The attack complexity is about $q^{m-\log_q(n-m)}$.

Example : Let $q = 16$ $m = 20$, $n \geq 37$. Then $k = 1$ and the attack works in 2^{76} instead of 2^{80}.

Another example : Let $q = 16$ $m = 16$, $n \geq 64$. Then $k = 1$ and the attack works in 2^{60} instead of 2^{64}.

This attack is never much better than the exhaustive search and we present better attacks below.

Remark : Let $h : (b_1, \dots, b_k) \mapsto (b_1, \dots, b_k, 0, \dots, 0)$. We chose an appropriate change of variables $U : (v_1, \dots, v_{m-k}) \mapsto (x_1, \dots, x_n$ that parameterizes X as $X = U(GF(q^{m-k}))$. Let $f' = f \circ U$.

We have reduced the problem of solving m quadratic equations with n unknowns to the problem of solving $m - k$ cubic equations in $m - k$ variables of the following form :

$$f'_{k+1..m}(v_1 + c_1 - f'_1(v), \dots,\ v_k + c_k - f'_k(v),\ v_{k+1}, v_{k+2}, \dots, v_{m-k}) \ = \ c_{k+1..m}.$$

8.7 The 'E' attack in $Max(q^{m/2}, q^{m-\sqrt{n/2}})$.

We chose k variables out of n and k' equations out of m. In this attack we do it at random, however in more advanced attacks (e.g. EFL) we will choose these subsets more carefully and make appropriate variable changes beforehand.

We write any equation l out of the subset $l = 1..k'$ as the following :

$$g_l(x_1, \dots, x_k) + \sum_{i=1..k} x_i \cdot (\sum_{k+1..n} \beta_{lij} x_j) + g'_l(x_{k+1}, \dots, x_n) = c_l$$

with g and g' multivariate quadratic. Our goal is to remove the part that mixes the $x_{1..k}$ and $x_{(k+1)..n}$, and in order to do that we need x that satisfies the kk' following linear equations

$$(\sum_{k+1..n} \beta_{lij} x_j) = 0.$$

Let $k = Min(m/2, \lfloor\sqrt{n/2-\sqrt{n/2}}\rfloor)$ and $k' = 2k$ (so that $k' \leq m$). Then we have

$$kk' \leq 2(\sqrt{n/2-\sqrt{n/2}})^2 \leq n - 2\sqrt{n/2} \leq n - 2k$$

Therefore $kk'+k \leq n-k$ and if we impose kk' equations on the $x_{k+1}, \dots, x_n$ we can still express them as a linear space of dimension at least k in new variables $x'_1, \dots, x'_k$. The equation to solve becomes :

$$g(x_1, \dots, x_k) + h(x'_1, \dots, x'_k) = c_{1..2k}$$

It is solved by birthday-paradox attack in q^k (Solving-in-th-Middle) instead of q^{2k}. Once this equation solved, with probability q^{2k-m} all the remaining $m - 2k$ equations will be satisfied too, if not we need to restart the whole attack described in this chapter with a different choice of initial variable subsets $\{1..k\}$ and equations subsets $\{1..2k\}$. Each such choice require about $(n-2k)^3$ computations, and it can be improved.

For simplification we assume that

$$(n-2k)^3 leqq^k$$

Then the **complexity of the attack is** q^{m-k} which by definition of k is

$$q^{Max\left(\frac{m}{2},\ m-\lfloor\sqrt{n/2-\sqrt{n/2}}\rfloor\right)}$$

Example 1 : Let $q = 16$, $m = 20$ and $n = 40$. Then $k = 4$ is suitable and the attack is in 2^{64} instead of 2^{80}.

8.8 The ‘FL’ attack in $q^{Max(m/2,\ m-\sqrt{2m})}$.

In this attack we fix $x_{k+1}, \dots, x_n$ and we get m quadratic equations in k variables which we regard as m linear equations in $(k+1)(k+2)/2-1$ variables x_i and x_ix_j. It is possible if

$$(k+1)(k+2)/2 - 1 \leq m$$

$$k \approx \lfloor\sqrt{2m+2} - 1.5\rfloor.$$

It seems that the attack require to do a Gaussian reduction on about m variables, the non-optimized complexity would be $\mathcal{O}(n^3)$. It can be reduced to $\mathcal{O}(k^3)$. Here is how proceeds the whole attack :

1. From the initial m equations we eliminate the linearized variables x_ix_j and we get k equations that have no quadratic terms in $x_{1..k}$. This is done only once. (We are now in the case of Special case Multi-Solving 2).
2. We fix some random values for the variables $x_{k+1}, \dots, x_n$.
3. We solve a system of k linear equations in $x_{1..k}$.
4. We go back to 2. as long as x does not satisfy the remaining q^{m-k} equations.

The complexity of the attack is about $q^{m-k} \cdot \mathcal{O}(k(n-k)^2)$ which is

$$q^{Max\left(\frac{m}{2},\ m-\lfloor\sqrt{2m+2}-1.5\rfloor\right)} \cdot \mathcal{O}(m^{3/2})$$

Example 1 : Let $q = 16$, $m = 20$ and $n \geq m$. Then $k = 5$ is suitable and the attack is in $2^{60} * 5^3 \approx 2^{67}$ instead of 2^{80}.

Example 2 : Let $q = 2$, $n = m = 80$. Then $k = 11$ and the attack is in $2^{69} * 11^3$ which is close to the exhaustive search in 2^{80}.

8.8.1 The 'FXL' attack.

This attack is the 'FL' attack for $d = 2$ and is probably never practical for $d > 2$ because of a growing polynomial factor in it's complexity.

This attack proceeds as follows :

$$k = \sqrt{d(d-1)m}, d \geq 2$$

- Fix $x_{k+1}, \ldots, x_n$
- Solve m quadratic equations in k variables by XL method [42]. The exact analysis of the XL method is not known but we estimated that it runs in $\mathcal{O}((k^d/(d)!)^3)$.

It appears that as for 'FL' the polynomial part complexity can be reduced to $\mathcal{O}((k^{d-1}/(d-1)!)^3)$ by the same method.

The complexity of the attack is (very roughly)

$$q^{Max(m/2,\ m-k)} \cdot \mathcal{O}((k/e)^{3d-3}), \quad \text{with } k = \sqrt{d(d-1)m}, \quad d \geq 2$$

8.9 The 'EFL' attack.

Among several ways we found to combine the 'E' and 'FL' attacks we might call EFL, it is unclear which version would give the best results.

We describe one possible version with a numerical evidence that it can be better than E or FL attacks alone.

It is our best attack in practice for solving multivariate equations in general.

1. As in 'E' we chose k variables out of n and k' equations out of m.

   We may choose any k' independent linear combinations of the initial equations. And we do it the way that no terms in variables $x_{k"+1}, \ldots, x_k$ does appear in the equations $1..k'$. It can be done as long as

   $$(k+1)(k+2)/2 - (k"+1)(k"+2)/2 \leq m - k' \quad (\mathbf{A}).$$

2. Now we proceed almost literally as in the algorithm 'E' : We restrict the linear space of x so that it is expressed as a linear space of $\lceil k'/2 \rceil$ new variables $x'_{1..\lceil k'/2 \rceil}$ and the first k' equations are of the form :

$$g(x_1,\ldots,x_k)+h(x'_1,\ldots,x'_{\lceil k'/2\rceil})=c_{1..k'}$$

In order to be able to do that we need :

$$kk'\le n-k-\lceil k'/2\rceil \quad (\mathbf{B}).$$

3. We pick $q^{\lceil k'/2\rceil}$ random values for $(x_1,\ldots,x_k)$ and of the form $(0,\ldots,0,x_{k"+1},\ldots,x_k)$. We can do it if

$$\lceil k'/2\rceil\le k-k" \quad (\mathbf{C}).$$

We also pick $q^{\lceil k'/2\rceil}$ random values $(x'_1,\ldots,x'_{\lceil k'/2\rceil})$. In $\mathcal{O}(q^{\lceil k'/2\rceil})$ we find a solution to the first k' equations. However since nothing depends on the first variables $x_1,\ldots,x_{k"}$, we have in fact found in $\mathcal{O}(q^{\lceil k'/2\rceil})$, a linear space of $q^{k"}$ solutions to the first k' equations.

4. Now we suppose that (from the beginning), in the remaining equations $k'+1..m$ the first variables $x_1,\ldots,x_{k"}$ has been eliminated as far as possible starting from m to $k'+1$. We consider a $k"'$ such that in the equations $k'+1,\ldots,k'+k"'$ the terms quadratic in $x_{1..k"'}$ are absent. We need to have

$$k"'(k"'+1)/2\le m-k'-k"' \quad (\mathbf{D}),$$

and it obvious that

$$k"'\le k \quad (\mathbf{E}).$$

Now we use the solution space found in 3., that does not depend on $x_{1..k"'}$, we substitute it to the equations $k'+1..k'+k"'$ and solve for $x_{1..k"'}$. Thus we get in $\mathcal{O}(q^{\lceil k'/2\rceil})\cdot\mathcal{O}(k"'^3)$, a linear space of $q^{k"-k"'}$ solutions to the first $k'+k"'$ equations.

5. With probability about $q^{k'+k"'-m}\cdot q^{k"-k"'}$ one of the $q^{k"-k"'}$ solutions will also solve the remaining $m-k'-k"'$ equations. If not, we repeat the whole attack 1..5 until it does. The number of repetitions is :

$$q^{m-k'-k"}$$

The complexity of the attack is

$$q^{m-k'-k"+\lceil k'/2\rceil}\cdot\mathcal{O}(k"'^3) \;=\; q^{m-k"-\lfloor k'/2\rfloor}\cdot\mathcal{O}(k"'^3).$$

The complexity of the attack is $q^{m-k"-\lfloor k'/2\rfloor}\cdot\mathcal{O}(k"'^3)$. It is unclear what are the best $(k,k',k",k"')$ choices.

The 'EFL' attack might still be improved using initial variable changes on the variables x_i that we have not used so far.

Example 1 : Let $q=16$, $m=20$ and $n=40$. Then $k=7, k'=4, k"=5, k"'=4$ and the attack is in $16^{13}\cdot 4^3=2^{58}$ instead of 2^{80}.

Example 2 : Let $q = 2$, $n = m = 80$. Then $k = 15, k' = 4, k" = 13, k"' = 10$ and if we see that a CPU can handle adding lines of a matrix over GF(2) in parallel using XOR operations, and therefore the Gaussian reduction is quadratic, our attack is in about $2^{65} \cdot 10^2 \approx 2^{72}$ instead of 2^{80}.

8.9.1 The 'EFXL' attack

In the same way as with FL and FXL, we can extend the previous attack and propose the EFXL attack. It is doubtful it can ever be practical.

8.10 Other attacks known for MQ with $n \gg m$

For finite fields of characteristic 2, there is another algorithm given in [76] that is polynomial when $n \geq m^2$. It is an open problem how to extend this result for fields of characteristic $\neq 2$.

Still for finite fields of characteristic 2, a new and different method has been recently proposed by Meier and Tacier in a work still in progress [45]. The complexity seems to be $q^{m-log_2(n/m)}$.

Chapitre 9

Algorithmes pour MQ avec un sous-corps

Cette partie est rédigée en anglais et montre une attaque qui casse de façon spectaculaire de nombreux cryptosystèmes multivariables, dès qu'on choisit des coefficients dans un sous corps de K.

9.1 Leçons à tirer

Nous allons ainsi casser dans 9.6 quatre schémas de signature qui nous ont été proposés par Jacques Patarin dans une première version de [7]. L'idée de prendre des coefficients des équations dans un sous corps de K est donc plus que dangereuse.

Cette attaque, nous amène à conseiller de prendre toujours une extension première K d'un corps premier $K = \mathbb{F}_{p^{p'}}$ dans tous les cryptosystèmes multivariables. Cela permet d'éviter l'existence même de sous-corps intermédiaires entre $\mathbb{F}_p$ et K.

9.1.1 Des attaques analogues

Dans le récent article [35] les auteurs montrent comment le choix des coefficients d'une courbe elliptique dans un sous corps réduit la complexité des attaques (bien que de très peu).

De même dans [126] Pierre Loidreau avait montré que prendre des coefficients du polynôme générateur d'un code de Goppa dans le sous corps de base $\mathbb{F}_2$ permet de cryptanalyser le cryptosystème de McEliece.

9.2 The MQ problem with a subfield.

Let $K = GF(q)$ and $K' = GF(p)$ be a subfield with $q = p^\theta$ (in this part p is not necessarily a prime).

Let f be a quadratic multivariate function $f : K^n \to K^m$ such that all the coefficients of quadratic terms of f are in the subfield K'.

(We do not require the linear or constant coefficients to be in K').

We may suppose that f has no constant terms, and the equation to solve is $f(X) = c$ with a constant $c \in K^m$.

We will use a representation of K as K'^θ such that $1 = (0, 0, \dots, 0, 1) \in K' \subset K$ and we write any vector $X_i \in K$ as $(X_{i1}, \dots X_{i\theta})$ with $X_{ij} \in K'$. If $X \in K^n$ we represent it as $n\theta$ elements of K' :

$$X = (\ (X_{11}, \dots, X_{1\theta}),\ \dots, (X_{n1}, \dots, X_{n\theta})\ ).$$

9.3 The attack framework.

Every element of K^n has an unique expression as

$$X + \alpha$$

with

$$\alpha = (\ (0, \dots, 0, \alpha_1),\ \dots, (0, \dots, 0, \alpha_n)\ ), \qquad \alpha_i \in K'$$

and with X of the form

$$X = (\ (X_{11}, \dots, X_{1(\theta-1)}, 0),\ \dots, (X_{na}, \dots, X_{n(\theta-1)}, 0)\ ), \qquad X_{ij} \in K'$$

Now we notice that

$$f(X+\alpha)_k - f(X)_k \quad = \quad f(\alpha)_k + \sum_{ij} p_{kij} X_i \alpha_j$$

with $p_{kij} \in K'$.

We pick a random X and we try to find an *alpha* that satisfies a part of each equation.

Once X is fixed, the problem of solving in *alpha* the equation $f(X+\alpha)_k = c$ amounts to solving $f(X+\alpha)_k - f(X) = c'$ which gives the following system of $m\theta$ equations over K' :

$$\begin{cases} f(X+\alpha)_{kl} - f(X)_{kl} = c'_l \\ \quad k = 1..m, \quad l = 1..\theta \end{cases}$$

We write it as the following :

$$\begin{cases} f(\alpha)_{kl} + (\sum_{ij} p_{kij} X_i \alpha_j)_l \quad = \quad c'_{kl} \\ \qquad k = 1..m, \quad l = 1..\theta \end{cases}$$

There are 2 things to note :

1. $f(\alpha)_k \in K'$, i.e. $\forall_{j \neq \theta} f(\alpha)_{kj} = 0$ and therefore quadratic terms in α_j will appear in $f(X+\alpha)_{kl}$ **only** for $l = \theta$.
2. Since $p_{kij} \in K'$ and $\alpha_j \in K'$

$$(\sum_{ij} p_{kij} X_i \alpha_j)_l \quad = \quad \sum_{ij} p_{kij} X_{il} \alpha_j.$$

9.4 The attack :

Once X fixed, we consider $f(X+\alpha)_{kl}$ as a function of α and each expression of

$$f(X+\alpha)_{kl}$$

is **linear** in α_j for all $k = 1..m$ and $l = 1\dots(\theta-1)$.

We have $m\cdot(\theta-1)$ linear equations with n variables α_j.

We can find a solution α that satisfies $\mathrm{Min}(n, m(\theta-1))$ of them (Gaussian reduction). The time to solve them is at most $\mathcal{O}(n^3)$.

Among the $m\theta$ equations (in K') of $f(X+\alpha) - f(X) = c'$, those that are not solved are

$$p^{m\theta - Min(n,m(\theta-1))} = p^{Max(m,m\theta-n)}.$$

The attack succeeds with probability

$$p^{-Max(m,m\theta-n)}.$$

Working condition : In this randomized attack, since we started choosing a random X, we need to repeat the attack $p^{Max(m,m\theta-n)}$ times, and in order to make it work we need that the number of possible X is at least :

$$p^{n\theta-n} \geq p^{Max(m,m\theta-n)}$$

Which is true if $\theta \geq 2$ and $n \geq m$.

9.5 Conclusion.

A system of m quadratic equations with n variables in $GF(p^\theta)$ such that the quadratic terms coefficient belong to the subfield $GF(p) \subset GF(p^\theta)$ with $\theta \geq 2$ is solved in less than

$$\mathcal{O}(n^3 * p^{Max(m,m\theta-n)}).$$

Important application note : The best results with this attack are sometimes achieved by weakening the assumptions. In order to minimize $(m\theta - n)$ and when the actual $\theta = dim_{K'}(K)$ is big, it may be interesting to consider that the coefficients are in a bigger subfield than the actual K'. We need to check for intermediate extension we still call K', such that $\theta = dim_{K'}(K)$ is the smallest possible with $\theta \geq \lceil \frac{n}{m} \rceil$.

The improvement described above works only if the initial θ is a composite. If θ is a **prime** big enough to have $m\theta >> n$, the attack will be impractical. This lead to choice of parameters for Sflash [61, 62] explained partly 9.6.3.

It is unclear if the attacks could not be improved used some bigger field that $GF(p^\theta)$ or using several subfields.

9.6 Application - breaking four signature schemes.

We cryptanalyse here some signature schemes proposed by Patarin for implementation in low-cost smart cards.
Important : we neglect the n^3 factor in the running time.

9.6.1 Application to Oil and Vinegar

Let's consider a function with 60 variables and 20 equations over $K = GF(16)$ with $K' = GF(2)$, $\theta = 4$.
The attack is in $2^{Max(20,20)} = 2^{20}$ instead of the exhaustive search in 2^{80}.

9.6.2 Application to another Oil and Vinegar

Let's consider a function with 60 variables and 20 equations over $K = GF(256)$ with $K' = GF(2)$.
It seems that the attack runs in $2^{Max(20,100)} = 2^{100}$. However we may consider that $K' = GF(4)$, $p = 4$, $\theta = 4$ and the complexity becomes $4^{Max(20,20)} = 2^{40}$ instead of the exhaustive search in 2^{160}.

9.6.3 Application to C^{*--} and Sflash

Sflash is an implementation C^{*--} with 37 variables and 26 equations over $GF(128)$ with the only possible $K' = GF(2)$. It s described in details in [61, 62]. Our current attack runs in $2^{Max(26,145)} = 2^{145}$ with the exhaustive search in 2^{182}. This attack played an important role in the design of Sflash. For this we suppose that we replaced $GF(128 = 2^7)$ by $GF(2^8)$:

A modified version of Sflash

We still consider a function with 37 variables and 26 equations over $GF(256)$ with $K' = GF(2)$ which we consider to be $K' = GF(16)$ with $\theta = 2$. The attack runs in $16^{Max(26,15)} = 2^{104}$ instead of the exhaustive search in 2^{208}.

9.6.4 Application to HFEv-

Let's consider a function with 38 variables and 32 equations over $GF(16)$ with $K' = GF(2)$ which we consider to be $K' = GF(4)$ with $\theta = 2$.
The attack runs in $4^{Max(32,26)} = 2^{64}$ instead of the exhaustive search in 2^{128}.

Chapitre 10

MQ en tant que problème difficile

10.1 Des instances difficiles de MQ

Il apparaît clairement avec l'algorithme XL que les instances difficiles de MQ sont dans la zone $m \leq n$.

Pour qu'un système ait en moyenne une solution, la propriété très utile en cryptographie, il faut que $n \approx m$.

Ainsi les cryptosystèmes multivariables, par exemple HFE, avec le plus souvent $m = n$ se placent dans cette zone.

Par exemple sur $\mathbb{F}_2$, avec $n = 80 - 128$, la meilleure attaque (y compris XL) pour résoudre MQ reste toujours (à peu près) la recherche exhaustive.

10.1.1 MQ et l'attaque de Courtois pour HFE

Quelques simulations pour comparer MQ aux attaques équationnelles sur HFE se trouvent dans 14.2. Ces attaques fonctionnent jusqu'à un certain n, assez petit.

10.2 MQ est NP-dur

MQ est NP-complet pour tout corps K, [11, 8, 7].

Nous donnerons ici une preuve dans le cas particulier de $K = \mathbb{F}_2$.

Preuve sur $\mathbb{F}_2$:

Il suffit de montrer que le problème 3-SAT, un des problèmes NP-complets de référence, se réduit polynomialement à MQ.

Les équations du problème 3-SAT consistent en m clauses qui sont des disjonctions d'au plus 3 littéraux. Chaque littéral est une variable booléenne ou la négation d'une variable.

Le problème 3-SAT consiste à décider si l'ensemble des clauses est satisfiable, c'est à dire s'il admet une solution.

D'abord on va écrire 3-SAT comme des équations cubiques. Le codage est simplement le passage de la forme normale disjonctive des fonctions booléennes à leur forme normale algébrique, par exemple :

$$\begin{cases} 1 = x \vee y \\ 1 = x \vee y \vee z \\ 0 = \neg t \\ \vdots \end{cases} \quad \begin{cases} 1 = xy + x + y \\ 1 = xyz + xy + yz + xz + x + y + z \\ 0 = 1 + t \\ \vdots \end{cases}$$

Ainsi on peut transformer une instance de 3-SAT avec m équations et n variables en un système de m équations de degré 3 avec n variables sur $\mathbb{F}_2$.

L'étape suivante consiste à donner un système quadratique équivalent au système donné de degré 3. Ceci se fait en ajoutant :

- nouvelles variables $y_{ij} = x_i x_j$.
- nouvelles équations **triviales** $0 = y_{ij} - x_i x_j$.

Lemme 10.2.0.1 (Codage efficace 3-SAT $\rightsquigarrow$ MQ) *Il suffit d'ajouter au plus m nouvelles variables, une par clause, au lieu de $n^2/2$ variables possibles de type $y_{ij} = x_i x_j$.*

Preuve : Il n'y a rien à faire pour les clauses contenant au plus 2 variables qui sont déjà de degré 2. Pour une clause contenant 3 variables la partie de degré 3 est toujours sous forme xyz. Il suffit d'ajouter une nouvelle variable $t = xy$ et une nouvelle équation $t = xy$.

Donc un 3-SAT avec m clauses et n équations donne une instance de MQ avec $2m$ équations et $m + n$ variables.

Remarque : Nous apprenons dans la littérature qu'un 3-SAT aléatoire n'est dur que si

$$m \approx 5.19n.$$

Cela donne un MQ avec environ $10 * n$ équations et $6 * n$ variables, et nos résultats sur XL (7.6) suggèrent que 3-SAT pourrait être sous-exponentiel en moyenne. On peut en douter car les instances de MQ ainsi générées ne sont pas aléatoires et peuvent se comporter de façon particulière. De plus on connaît déjà des algorithmes très efficaces pour 3-SAT jusqu'à environ 5000 équations [47]. La complexité de XL pour de telles valeurs de n sera certainement beaucoup plus grande. On ne peut donc même pas à l'heure actuelle vérifier expérimentalement si 3-SAT est sous-exponentiel avec FXL.

10.3 MQ et les fonctions à sens unique

MQ est un candidat naturel pour une fonction à sens unique : il est NP-complet, il effectue les opérations non-linéaires sur les bits, et en même temps le cardinal des fonctions possibles est assez grand, de l'ordre de $q^{mn^2/2}$. Prendre des fonctions de degré ≥ 3 ne pas intéressant, car la taille de la description de la fonction croît très vite, et il faudra rapidement des Giga octets pour stocker une telle fonction.

Dans la littérature on trouve des nombreuses notions de fonctions sens unique plus moins fortes. MQ ne les satisfait pas toutes. Par example :

1. OWF (One-Way function) - fonction à sens unique ordinaire - **Oui**, si $P \neq NP$.
2. SPR (Second-Preimage Resistant) - Pb. ouvert, En pratique il semble que Oui.
3. WUF, UOWHF (Universal One-Way Hash Function), ou Target-Collision Resistant - Pb. ouvert, En pratique il semble que Oui.
4. CR (Collision Resistant), fonction sans collision - **Non**.
5. PRF (Pseudo-Random Function), **Non** car ce n'est pas une CR.

10.3.1 MQ n'est pas sans collisions (CR)

Il est facile de montrer que MQ n'est pas sans collisions pour $m \leq n$. En effet, pour tout z fixé l'équation $f(x+z) - f(x) = 0$ est linéaire en les x_i et on trouvera les solutions par la réduction de Gauss sur m équations avec n inconnues avec $m \leq n$.

Même si $m > n$ nous pouvons trouver une collision avec une probabilité de q^{m-n}. Le problème de savoir s'il est difficile de trouver une collision sur MQ avec $n^2 >> m > n$ mieux qu'en q^{m-n}, et mieux qu'en utilisant la simple inversion de la fonction, reste ouvert.

Par contre le problème de trouver les collisions sur MQ avec $m >> n$ est facile et résolu un utilisant l'inversion par XL ou linéarisation, voir le chapitre 6.2.4.

10.3.2 Le sens unique, MQ et la clef publique

Nous avons vu que certaines notions de sens unique connue s'appliquent à une fonction MQ, d'autres pas. Dans la suite du présent manuscript des nombreuses applications de MQ en cryptographie à clef publique sont étudiées.

Nous soutenons que la notion de sens unique pour une fonction trappe n'est plus la même et qu'il y a besoin des notions plus adaptées. Sans prétendre d'apporter une réponse définitive, nous allons expliquer comment on peut introduire de telles notions ainsi quel est leur intérêt.

10.3.3 Vers une notion algébrique de fonction à sens unique

Définition [très informelle] : Fonction à sens unique par rapport à une mesure de complexité :

C'est une fonction qui **ne permet** d'obtenir par *des opérations de base* (algébriques) et avec une probabilité non négligeable, aucune équation **explicite**, ni aucune équation **implicite** strictement plus simple par rapport à la mesure de complexité donnée.

10.3.4 Définir 'des opérations de base'

Soient $g(x) = y$ les équations à résoudre, que l'on réécrit sous forme suivante :

$$\begin{cases} l_1 &= 0 \\ &\vdots \\ l_n &= 0 \end{cases}$$

Définition [informelle] : Des opérations de base algébriques :

C'est toutes les opérations qui permettent d'obtenir d'autres équations qui :
- sont des combinaisons polynomiales **explicites** des équations données l_i avec des variables x_i.
- sont de 'petit' degré (ou de 'petite' taille)
- sont égales à 0 avec la probabilité 1.

Exemple : $x_1 l_5 + x_2 l_1 + l_3 l_5 = 0$, le degré est 4.

Menace : Que se passe-t-il si on substituait, par exemple les l_i avec leur expressions en x_i, on obtiendrait par exemple :

$$x_1 l_5 + x_2 l_1 + l_3 l_5 \;=\; x_1,$$

Et comme tous les l_i sont des expressions en les x_i qui s'annulent en la valeur x recherchée, on a :

$$0 \;=\; x_1.$$

On voudrait que cela n'arrive jamais, et que les combinaisons obtenues de façon triviale ne permettent pas d'obtenir soudainement d'équations non-triviales sur des x_i.

On voudrait qu'elles restent 'triviales' quant à leur propriétés (p.ex. degré, taille, le nombre de variables impliqué ou d'autres mesures de complexité).

10.3.5 La notion d'équations triviales

Définition [informelle] : Équation triviale (par rapport à un mesure de complexité) :
- C'est une somme de produits de petit degré des l_j et x_i
- dont chacun contient au moins un l_j, et tels que
- après la substitution $l_k = \sum x_i x_j ...$, la complexité de l'équation par rapport à une mesure de complexité donnée (p.ex. le degré multivariable) ne s'effondre pas.

Le contraire mène directement à des attaques.

10.3.6 Applications des équations non-triviales

Substituer $l_j \equiv 0$,ce qui donne de **nouvelles** équations de petite complexité (p.ex. de petit degré) sur les x_i.

On peut itérer cette attaque.

10.3.7 Le cryptanalyste automatique

Ce type d'attaque marche parfois sans qu'on puisse savoir pourquoi.

Les meilleures attaques connues de HFE, [Courtois 99], décrites dans 13.3.2 et [68, 70] sont de ce type (avec des améliorations techniques).

10.4 Codage de la factorisation dans MQ

Cette brève partie est rédigée en anglais. Son but est simplement d'évaluer le nombre d'équations de degré 2 nécessaires pour coder complètement le problème de factorisation d'un entier de 512 bits, ce qui peut servir à créer des instances dures de MQ sparse.

Factoring with (sparse) quadratic equations.

We consider the problem of factoring an integer z of n bits, $z = z_0, \ldots, z_{n-1}$.

We assume that x and y are two factors of $n/2$ bits each. We consider a traditional pencil and paper multiplication method :

- For every i such that $y_i = 1$ we write in a separate line the binary x shifted i positions to the left.
- We add all the lines and we should get z.

Let $c = (c_0, .., c_{n-1})$ be the carry values in this addition. Since in every column we are adding at most $n/2$ bits, we may as well assume we are working in $GF(p)$, $p > n/2$. We get n following equations over $GF(p)$:

$$\forall_{k=0..n-1} \qquad 2c_{k+1} + z_k \;=\; \sum_i y_i x_{k-i} + c_k \text{ with } \; c_0 = 0 \; \text{ and } \; c_n = 0.$$

We re-write these equations with the individual bits $c_{ij}, j = 0.. \log n$ of every c_i :

$$\forall_{k=0..n-1} \qquad \sum_{j=0..\log n} c_{k+1,j} 2^j + z_k \;=\; \sum_i y_i x_{k-i} + \sum_{j=0..\log n} c_{k,j} 2^j$$

with $c_0 = 0$ and $c_n = 0$.

We add n equations that express the fact that the x_i and the y_i are 0 or 1 :

$$\forall_{k=0..n/2} \qquad \begin{cases} x_k^2 &= x_k \\ y_k^2 &= y_k \end{cases}$$

And we also add $n \log n$ equations that say that the c_{ij} are 0 or 1 :

$$\forall_{i=0..n-1} \forall_{j=0..\log n} \qquad \left\{ \; c_{ij}^2 \;=\; c_{ij} \right.$$

It is easy to see that factoring is equivalent to the problem of solving these $2n + n \log n$ quadratic equations with $n + n \log n$ variables over $GF(p)$.

In practice, in order to factor 512-bit numbers we need to solve a system of 5632 quadratic equations with 5120 unknowns.

Moreover all these equations are sparse and most of them are composed of only 2 terms.

Quatrième partie

Le problème HFE

Chapitre 11

Préliminaires

11.1 Rappels sur les corps finis, suite

A la suite du chapitre 5.1 on posera toujours $K = \mathbb{F}_q$. En pratique on a assez souvent $K = \mathbb{F}_2$. La construction d'une extension d'un corps fini décrite dans 5.1.3 peut être appliqué de façon analogue à K lui même.

La construction de $K^n = \mathbb{F}_{q^n}$

- $\mathbb{F}_q[X] =$ est l'ensemble de polynômes en X avec les coefficients dans $\mathbb{F}_q$.
- Soit P un polynôme irréductible de degré n sur $\mathbb{F}_q$.
- $\mathbb{F}_{q^n} \stackrel{def}{=} \mathbb{F}_q[X]/P(X)$, c'est l'ensemble des polynômes de $\mathbb{F}_q[X]$ modulo $P(X)$.
- $\mathbb{F}_{q^n}$ est une extension du corps (de base) $\mathbb{F}_q$. C'est aussi un espace vectoriel de dimension n sur $GF(q)$:
  Tout élément $x \in \mathbb{F}_{q^n}$ peut être codé comme les n coefficients d'un polynôme $x \in K[X]/P(X)$.

11.1.1 Les représentations univariables et multivariables

On a vu que tout élément x de $\mathbb{F}_{q^n}$ peut être vu alternativement comme n éléments de $\mathbb{F}_q$. On identifie $K^n = \mathbb{F}_{q^n}$.
Si $x = x_1 + x_2 \cdot X + \ldots + x_n \cdot X^{n-1}$,
alors on écrit $x = (x_1, \ldots, x_n)$.

Les fonctions ont également deux représentations, **Toute** fonction $f : K^n \rightarrow K^n$ s'écrit comme :

1. un polynôme univariable f sur K^n.
2. n équations multivariables f_i sur K.

On écrira donc indifféremment :

$$f(x) \;=\; f(x_1, \ldots, x_n) \;=\; (f_1(x), \ldots, f_n(x)).$$

Par contre le degré univariable et multivariable d'une fonction ne coïncident pas.

11.1.2 Degré univariable/multivariable

Les propositions suivantes sont faciles à démontrer, une preuve constructive est contenue dans [71].

1. Une fonction de forme $b = f(a) = a^{q^s}$ donne dans la représentation multivariable $b_i = f_i(a_1, \dots, a_n)$, avec les f_i de degré 1 sur K.
2. Une fonction s'écrit sous la forme $f(a) = \sum_i \lambda_i a^{q^{s_i}}$, $\lambda_i \in GF(q^n)$, si et seulement si tous les f_i sont homogènes de degré 1 sur K.
3. Si $f(a) = a^{q^s+q^t}$, les f_i sont de degré 2 sur K.
4. Soit q un nombre impair.
   Une fonction s'écrit sous la forme $f(a) = \sum_i \lambda_i a^{q^{s_i}+q^{t_i}}$, $\lambda_i \in GF(q^n)$, si et seulement si tous les f_i sont homogènes de degré 2 sur K.
5. Une fonction s'écrit sous la forme $f(a) = \sum_i \lambda_i a^{q^{s_i}+q^{t_i}} + \sum_i \lambda_i a^{q^{s_i}} + C$, $\lambda_i \in GF(q^n)$, si et seulement si les f_i sont (non-homogènes) de degré ≤ 2 sur K, dont au moins une est de degré 2.

Exemple

Soient $K = \mathbb{F}_2 = \{0, 1\}$, $n = 3$
$P(X) = X^3 + X^2 + 1$ est irréductible modulo 2.
On a $K^n = \mathbb{F}_{2^3} = K[X]/X^3 + X^2 + 1$.
On dérivera la représentation multivariable de la fonction suivante :
On écrit $a \in \mathbb{F}_{2^3}$ sous la forme $a_2X^2 + a_1X + a_0 \in K[X]/P(X)$.
$b = f(a) = a + a^3 + a^5 =$
$(a_2X^2 + a_1X + a_0) + (a_2X^2 + a_1X + a_0)^3 +$
$(a_2X^2 + a_1X + a_0)^5 \ mod \ X^3 + X^2 + 1 \ = \ (...) \ =$
$(a_2 + a_2a_1 + a_2a_0 + a_1)X^2 + (a_2a_1 + a_1a_0 + a_2)X + (a_0 + a_2 + a_1a_0 + a_2a_0)$
Ce qui donne 3 équations quadratiques :

$$\begin{cases} b_2 &= a_2 + a_2a_1 + a_2a_0 + a_1 \\ b_1 &= a_2a_1 + a_1a_0 + a_2 \\ b_0 &= a_0 + a_2 + a_1a_0 + a_2a_0 \end{cases}$$

Chapitre 12

HFE et problème HFE

12.1 Le problème HFE

Le nom de HFE correspond à un problème mathématique bien défini sur des corps finis. Il a été pour la première fois posé par Jacques Patarin à Eurocrypt'96 [65]. Le problème HFE est défini pour un triplet K, n, d, avec $K = \mathbb{F}_q$ un corps fini, $n \in \mathbb{N}$ la taille des blocs, et avec $d \in \mathbb{N}$ qui est un paramètre de sécurité qui sera le degré maximum du polynôme caché (**Hidden** Field Equation). Dans la définition qui suit on identifie le corps fini $K^n = \mathbb{F}_{q^n}$ et l'espace vectoriel K^n.

Le polynôme caché de HFE

Soit $f : K^n \to K^n$, $K = \mathbb{F}_q$ définie comme :

$$f(a) \stackrel{def}{=} \sum_{q^s+q^t \leq d} \gamma_{st} \; a^{q^s+q^t}$$

Soient S et T deux changements de variables affines $S, T : K^n \to K^n$.

La fonction HFE.

Appliquer S et T sur la représentation multivariable de f :

$$g \stackrel{def}{=} T \circ f \circ S.$$

Le problème HFE consiste à inverser g en un point aléatoire avec une probabilité de succès non-négligeable, plus précisément :

Le problème HFE avec (T, ε)

Pour un $y \in K^n$ uniformément choisi au hasard, trouver en temps T, et avec une probabilité $\geq \varepsilon$, au moins un inverse $x \in K^n$ tel que $y = g(x)$.

Le problème HFE ainsi décrit correspond au problème de l'inversion de ce qui est appelé 'basic' HFE dans [65].

12.1.1 Pourquoi HFE

HFE veut dire **H**idden **F**ield **E**quation. C'est un magnifique exemple d'ambiguïté syntaxique dans la langue anglaise. Il y a $3 = \binom{3}{2}$ traductions possibles dont chacune correspond à une de $\binom{3}{2}$ façons de relier 2 mots parmi 3 en une unité de sens, puis à la fin on ajoute le troisième mot. Ainsi on peut traduire *Hidden Field Equation* par, au choix comme :

1. *Hidden (Field) Equation* L'équation cachée sur un corps.
2. *(Hidden Field) Equation* L'équation sur un corps caché.
3. *Hidden (Field Equation)* L'équation d'un corps caché.

Chacune de ces trois traductions correspond à une de trois façon de voir le problème HFE, toutes aussi légitimes.

12.1.2 Problèmes annexes

Il ne faut pas confondre le problème HFE et la cryptanalyse de HFE. Le cryptosystème HFE admet des variantes qui s'avèrent solides malgré toutes les attaques sur le problème algébrique HFE. De même toutes les attaques connues échouent contre ces variantes, elles sont décrites dans 17.1 et [7, 65, 8].

Il faut aussi distinguer le problème de récupérer la clef secrète de HFE, qui se réduit comme on le verra plus tard au problème MinRank [71]. Bien des cryptosystèmes à clef publique ont été cassés sans récupérer la clef secrète. Il s'agit donc ici uniquement du problème d'inversion de la fonction de chiffrement. Cela est en principe plus facile que de récupérer la clef secrète, et donc la difficulté du problème HFE impliquera à fortiori qu'il est difficile d'obtenir la clef secrète.

Enfin, le problème IP décrit dans 19 et [65, 88] donnerait une attaque de HFE uniquement si f était donnée, ce qui n'est pas le cas dans HFE.

12.2 Le problème HFE en chiffrement

Son utilisation est directe (avec notations de la page précédente) :

Clef publique : La clef publique est g qui n'est pas donnée sous forme originale univariable $g \stackrel{def}{=} T \circ f \circ S$, mais exprimée en tant que n équations quadratiques avec n variables.

Clef secrète : Deux transformations affines T, S, ainsi que f donné sous forme univariable de degré $\leq d$.

Inversion Pour inverser g on utilise la formule

$$g^{-1} \ = \ S^{-1} \circ f^{-1} \circ T^{-1}.$$

Et comme f est de degré borné, on peut calculer f^{-1} par les méthodes univariables, par exemple l'algorithme de Berlekamp ou autre [3, 65, 70]. Ces algorithmes, comme on le verra dans 17.2.1 ne sont pas très rapides.

Dans 2.4 j'explique pourquoi et comment, le problème HFE suffit à lui même pour construire des schémas de chiffrement prouvés sûrs (!).

12.2.1 Example détaillé d'utilisation de HFE en chiffrement

Pour clarifier les choses on donne ici un petit example de HFE avec $K = \mathbb{F}_2 = \{0,1\}$, $n = 80$, $d = 96$. Ces paramètres correspondent au HFE Challenge 1 publié dans [65, 70] et également étudié dans 13.1 et 16.2.

Soit $\mathcal{L} = F_2[X]/(X^{80} + X^{38} + X^{37} + X + 1)$. Pour être le plus explicite possible, on va noter φ la bijection canonique entre $K^{80} = \{0,1\}^{80}$ et $\mathcal{L} = \mathbb{F}_{2^{80}}$ (omise dans la description précédente) avec :

$$\varphi(\omega) = \omega_{79}X^{79} + \ldots + \omega_1 X + \omega_0 \pmod{X^{80} + X^{38} + X^{37} + X + 1}.$$

On notera F un polynôme secret $F : \mathcal{L} \to \mathcal{L}$ de degré $d = 96$,

$$F(X) = \sum_{\substack{0 \le i < j < 80 \\ 2^i + 2^j \le 96}} \alpha_{i,j} \cdot X^{2^i + 2^j} + \sum_{\substack{0 \le i < 80 \\ 2^i \le 96}} \beta_i \cdot X^{2^i} + \gamma.$$

Avec les coefficients $\alpha_{i,j}$, les β_i ainsi que la constante γ pris aléatoirement dans $\mathcal{L} = \mathbf{F}_{2^{80}}$. Compte tenu du fait que les puissances présentes sont les sommes de une ou deux puissances de 2, on a des exposants suivants dans F : $X^0, X^1, X^2, X^3, X^4, X^5, X^6, X^8, X^9, X^{10}, X^{12}, X^{16}, X^{17}, X^{18}, X^{20}, X^{24}, X^{32}, X^{33}, X^{34}, X^{36}, X^{40}, X^{48}, X^{64}, X^{65}, X^{66}, X^{68}, X^{72}, X^{80}, X^{96}$.

Grâce à cette forme particulière des puissances présentes dans F, et comme expliqué dans 11.1.2, on peut écrire F en tant que n équations quadratiques avec n variables $x_i \in \{0,1\}$. On notera f cette version de F ré-écrite en tant que n équations quadratiques avec n variables $x_i \in \{0,1\}$. En fait il suffit de poser :

$$f \stackrel{def}{=} \varphi^{-1} \circ F \circ \varphi$$

En pratique il est possible de calculer formellement la version multivariable de F, ou la récupérer à partir des valeurs de F en un certain nombre de textes clairs de petit poids. Ensuite on applique deux changements de variables affines S et T dans chacun des $n^2 + n$ coefficients est choisi au hasard dans $K = \{0,1\}$. Ensuite on pose

$$g \stackrel{def}{=} T \circ f \circ S = T \circ \varphi^{-1} \circ F \circ \varphi \circ S$$

La clef publique : C'est g. Pour la décrire on a besoin de donner tous des coefficients des n équations quadratiques avec n variables dans $K = \{0,1\}$. La taille de la clef publique est donc $n^3/2 \cdot \log_2 q$ bits, c'est à dire 32 Ko.

La clef secrète : Elle se compose de S, T, f. Les deux transformations affines T, S nécessitent $2(n^2 + n) \log_2 q$ bits. f doit être donné sous forme univariable, ce qui demande $Dn \log_2 q$ bits, D étant le nombre de coefficients égal à 28 si l'on prend F monique.

Inversion de la fonction trappe g Pour inverser g on utilise la formule

$$g^{-1} = S^{-1} \circ \varphi \circ F^{-1} \circ \varphi^{-1} \circ T^{-1}.$$

Pour inverser F on utilise une des méthodes connues qui factorisent des polynômes sur des corps finis. Soit $F^{-1}(b)$ la valeur à calculer. On va factoriser le polynôme $((F(a) - b))$ qui est de degré 96 à coefficients dans $\mathcal{L}$. Cela prend 1.260 secondes avec un Pentium III à 500 MHz. Notre programme en C++ qui donne ce résultat, est téléchargable à `http://www.minrank.org/hfecode.txt`. Il utilise des fonctions de la librairie NTL de Victor Shoup [39] qui avait été conçue dans le but de factoriser des polynômes.

Assez souvent il y aura plusieurs racines. Le nombre des racines suit en général la loi $P(i) = \frac{1}{ei!}$ expliquée à la page 148. Cela oblige à ajouter de la redondance au clair pour déchiffrer les messages de façon unique. Une autre façon de procéder est d'ajouter en plus du message chiffré $g(m)$ un haché $H(m)$ par une fonction de sens unique. Cette première solution est préférable car elle ne nécessite pas d'implementer une fonction de hachage (ni de supposer qu'elle est solide). Il est également conseillé d'ajouter au clair un certain nombre de bits aléatoires. Ceci permet d'éviter les attaques de type dictionnaire sur m et supprime la possibilité de créer des chiffrés valables sans connaître le clair correspondant.

Sécurité La meilleure attaque connue contre cet example est décrite dans la présente thèse et d'après 16.2 nécessite 2^{62} de calculs et 390 Go de mémoire.

12.3 Le problème HFE en signature

Le problème de la sécurité (théorique) des schémas de signature est traité dans le chapitre 2.5. Les applications pratiques de HFE en signature sont présentées dans 17.4. Un example connu d'utilisation de HFE en signature est Quartz [66]. Dans [69] l'auteur décrit un schéma novateur basé sur HFE qui permet de faire de la signature mais ne permet pas de faire du chiffrement.

12.4 Les choix de paramètres de HFE

En général le choix de paramètres pour une application de HFE n'est pas aisé. Tout d'abord le couple q, n doit éviter les attaques sur MQ décrites dans les chapitres 6 à 9, tout en ayant une taille de clef publique acceptable. Ensuite le paramètre de sécurité d du cryptosystème HFE doit être choisi le plus grand possible pour éviter les attaques décrites dans les chapitres 13 à 16, mais trop grand pour ne pas ralentir le déchiffrement. Voir des examples donnés dans 17.3.1 et 17.4.

Chapitre 13

La cryptanalyses de HFE

Dans cette partie on va décrire **tous les attaques connus** pour cryptanalyser le cryptosystème HFE, à l'exception des attaques génériques sur MQ qui ne prennent pas en compte l'existence d'une trappe dans HFE, traités dans 6, et à l'exception des attaques spécifiques sur les schémas de signature, qui n'utilisent pas non plus la trappe, et qui sont étudiés dans 18.

Dans la partie suivante on présente des simulations effectuées. Les résultats des attaques basées sur ces simulations sont décrits dans 15. D'autres attaques plus avancées seront décrits par la suite dans 15.2. Enfin la partie 16 va résumer la situation.

13.1 Des challenges de référence

Il existe deux challenges sur HFE pour la somme de $\boxed{2 * 500\ \$}$, [70, 8, 7].

Seul le premier de ces deux challenges est un pur problème algébrique HFE (i.e. un 'basic HFE'). Il s'agit d'une fonction trappe $g : \mathbb{F}_2^{80} \rightarrow \mathbb{F}_2^{80}$ qui sera notre point de référence pour comparer de différentes attaques. Le degré de HFE dans le Challenge 1 est $d = 96$ et $n = 80$.

13.2 Attaques sur la clef secrète

13.2.1 Comment trouver S et T.

Si f est connue, c'est le problème **IP** défini dans 19 et [65, 88, 7, 8].

Il existe une attaque en $q^{n/2}$ [Courtois, Eurocrypt'98] décrite dans 19.3.

Par contre l'essentiel de l'attaque de Shamir-Kipnis décrite dans ce qui suit, est de remarquer que dans une représentation spéciale on peut dire que f est dans un certain sens connu à 99%, car $d << q^n - 1$ qui est le degré maximal pour une fonction univariable.

13.2.2 L'approche univariable

Shamir avait été le premier à caractériser ainsi le problème de trouver la clef secrète de HFE, Crypto'99 [71].

Pour simplifier on se restreindra aux fonctions f, g homogènes de degré 2.

Représentation matricielle univariable Les fonctions quadratiques g et f peuvent être écrites dans une représentation univariable sous forme d'une matrice symétrique G telle que :

$$g(x) \; = \; \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} G_{ij} \; x^{q^i+q^j}$$

Rank(G) est a priori maximal ce qui a été confirmé par nos expériences.

Rank(G)= n

Par contre Rank(F)= r avec $r = \log_q d$ puisque le degré de f est borné par d.

Caractériser la faiblesse de HFE consiste à exprimer le fait que le rang s'effondre sous un certain changement de variables.

13.2.3 La technique de Shamir

On cherche S et T tels que :

$$T^{-1} \circ g \; \stackrel{?}{=} \; f \circ S$$

On expliquera ici comment trouver T. Ensuite, il sera assez facile de récupérer S, nous ignorerons cette partie décrite dans [71] et aussi dans [88] (résolution de IP avec un seul secret).

Lemme 1 [Shamir] : La représentation matricielle G' de $f \circ S$ est sous la forme $G' = WGW^t$, avec W est inversible.

Corollaire : G' a le même rang r que G.

$$Rank(G') = r$$

Lemme 2 [Shamir] : Soit t_i les coefficients de la représentation non standard de T^{-1}. La représentation matricielle de $T^{-1} \circ g$ s'écrit comme une combinaison linéaire de représentations matricielles de g^{q^k} notées G^{*k}. Donc on a

$$G' = \sum_{k=0}^{n-1} t_k G^{*k}$$

13.2.4 Exprimer HFE comme MinRank.

Ainsi comme $T^{-1} \circ g = f \circ S$, le problème de trouver T s'écrit :

$$Rank(\sum_{k=0}^{n-1} t_k G^{*k}) \leq r$$

Comme en général dans HFE d est au pire polynomial en n, afin de pouvoir inverser f dans 12.2, on a $r = \log d = \mathcal{O}(\log n)$.

Shamir a réduit le problème HFE au problème de rang minimal pour des matrices $n \times n$ avec des coefficients dans K^n. Ce problème s'appelle MinRank et cette instance particulière est notée MinRank$(n, n, n, \log_q n, GF(q^n))$ en suivant

les notations de 22.1. Pour l'instant, les seules attaques connues sur la clef secrète de HFE sont basées sur ce problème algébrique MinRank.

MinRank est NP-complet. Cela est vrai quand $r = n^{\varepsilon}, \varepsilon > 0$, voir 23.1 et [112]. Par contre il ne l'est probablement pas quand le rang est logarithmique, comme c'est le cas pour le MinRank obtenu ici avec HFE. En effet, dans les deux chapitres suivants on montre des méthodes pour résoudre ce problème en temps sous-exponentiel.

Applications. Cette réduction de HFE au problème MinRank admet deux utilisations possibles en cryptanalyse, qui correspondent à 2 façons de résoudre le problème MinRank, et qui sont décrites dans les deux chapitres qui suivent.

13.2.5 Réduction de MinRank à MQ

L'algorithme qui réduit MinRank à MQ est décrit dans 24.5.1 et [71].

Il écrit des équations qui disent que chacune des colonnes $[r+1, \ldots, n]$, est une combinaison linéaire des colonnes $[1, \ldots, r]$.

Le MinRank donne $n(n-r)$ équations quadratiques avec $r(n-r)$ variables sur K^n que l'on résout par la suite avec XL.

Soit $2 \leq \omega \leq 3$ l'exposant de la réduction de Gauss. La complexité de l'attaque, en suivant 24.5.1, vaut

$$(rn)^{\omega r^2} \approx \left(n \log_q d\right)^{\omega \log_q^2 d} \approx n^{\mathcal{O}(\log_q^2 d)}$$

Attaque de HFE par Relinéarisation

L'article de Shamir-Kipnis [71] propose en effet d'écrire ce MinRank comme MQ avec $n(n-r)$ équations et $r(n-r)$ variables sur K^n.

Shamir et Kipnis ont réduit successivement :

HFE$\rightsquigarrow$MinRank$\rightsquigarrow$MQ.

Ensuite on propose de résoudre ces équations par relinéarisation décrite dans 6.3.

Sachant que XL est une version améliorée de le relinéarisation, voir 6.3, nous avons appliqué la complexité de XL à l'attaque de Shamir-Kipnis.

Cela donne des chiffres astronomiques, par exemple environ 2^{152} pour le Challenge 1 sur 80 bits tandis que la recherche exhaustive est en 2^{80}.

13.2.6 Attaque par MinRank et de sous-matrices

La méthode améliorée, proposée par Courtois, a déjà été utilisée avant par Coppersmith, Stern et Vaudenay dans [77, 78] et est également étudiée en toute généralité dans 24.5.2 et [68]. L'algorithme est particulièrement simple et l'exprime directement MinRank comme des déterminants de sous-matrices.

On écrit les équations de type

$$0 = \text{dét(tous les mineurs } (r+1)\text{x}(r+1) \text{ de } G')$$

Ensuite on peut les résoudre par simple linéarisation. Comme $r = \log_q d$ et d'après 24.5.2, cette attaque donne la complexité de

$$\left(\frac{ne}{\log_q d}\right)^{\omega(\log_q d+1)} \approx n^{3 \log_q d}$$

Application à HFE donne environ 2^{97} pour le Challenge 1 sur 80 bits.

13.3 Attaques sur l'inversion

13.3.1 Faut-il récupérer la clef publique ?

La question est pertinente, car en effet en cryptographie à clef publique il arrive souvent de casser un schéma sans récupérer la clef publique et il en est de même dans le domaine multivariable.

1. La cryptanalyse de **certains** schémas donne la clef secrète :
   - D^* [Courtois 97].
   - 'Balanced Oil and Vinegar' [Kipnis, Shamir Crypto'98]
   - HFE [Kipnis, Shamir Crypto'99].
2. **Beaucoup** sont cassés sans récupérer la clef secrète.
   - 2 schémas de Shamir. [Stern, Coppersmith, Vaudenay]
   - Matsumoto et Imai C^* et $[C]$ schemes [Patarin]
   - D^*, Little Dragon, S-boxes, C^{*-} [Patarin, Goubin, Courtois]

Note 1. L'auteur du présent manuscrit a été le premier à réussir à récupérer réellement la clef secrète d'un schéma multivariable pour D^* et C^* [58].

Note 2. Cela est également l'occasion de voir d'autres schémas avec des polynômes multivariables existants qui ne sont que peu étudiés dans la présente thèse alors que de nombreux de nos attaques s'y appliquent.sur plusieurs schémas. Pour en savoir plus sur ces schémas voir 2.7 et surtout [7, 8].

13.3.2 Attaques équationnelles [Patarin, Courtois]

L'avantage de l'attaquant en cryptographie à clef publique (déterministe), par rapport à la cryptographie à clef secrète, consiste à ce qu'il est possible de manipuler les équations publiques g_j.

La notion suivante, déjà introduite dans 10.3.5 joue un rôle central dans toutes les attaques à venir :

Définition [informelle] : Une équation triviale est une combinaison de petit degré des g_j et des x_i, dont tous les termes contiennent au moins un g_j, et dont le degré multivariable ne s'effondre pas.

Le bien fondé : Une équation triviale substituée avec les $y_j = 0$ donne une nouvelle équation de petit degré en les x_i. On peut itérer l'attaque jusqu'à trouver les équations linéaires.

Notons que cette notion est très relative, car aussi bien le degré a priori (attendu), par rapport au degré à posteriori des équations (réel) ne dépend pas des équations elle-mêmes mais de la façon dont elles étaient obtenues.

Par exemple, les $x_i g_j$ auront un degré a priori 3, mais certaines de leurs combinaisons linéaires peuvent avoir degré 2 ou 1. Cela arrive dans les simulations qui suivent pour les cas HFE de degré ≤ 8.

13.3.3 Types d'équations

Nous avons une notation pour décrire un type d'équations par l'ensemble de leur termes.

1. On utilise les lettres $\{x, y, X, Y\}$, par exemple : $XY \cup x^2$. On peut alternativement utiliser $\{x, l, X, L\}$.
2. On dénote 1 l'ensemble de termes constants possibles.
3. Soit $x^k y^l$ l'ensemble de tous les termes de degré **exactement k** en les $x_i, i = 1..n_a$ et de degré **exactement l** en les $y_i, i = 1..n_b$. Il convient de noter que pour les termes dans $K = \mathbb{F}_q$ les degrés sont réduits modulo q.
4. Le symbole $\cup$ dénote l'union habituelle des ensembles.
5. Les majuscules X, Y signifient que l'on inclut aussi des termes de degré inférieur. Par exemple $XY \cup x^2$ est synonyme de $1 \cup x \cup y \cup xy \cup x^2$.
6. On note $size_T$ le cardinal de l'ensemble T. Par exemple $size_{X \cup Y} = m + n + 1$.
7. La valeur de $size_T$ diffère en fonction de la caractéristique p de K.
   Par exemple, si $p = 3$ on a $x_2^2 \in \{x^2\}$ et $size_{x^2} = n(n-1)/2$.
   Par contre si $p = 2$, on a $x_2^2 \notin \{x^2\}$ et $size_{x^2} = n(n+1)/2$.

13.3.4 La taille des équations

Le tableau suivant présente quelques tailles $size_T$ que nous avons calculées avec de la combinatoire élémentaire :

eqn. type	1	x	y	xy	x^2	x^2y	y^2
size	1	n	m	$n*m$	$n(n-1)/2$	$m*n(n-1)/2$	$m(m-1)/2$
+if $p > 2$	0	0	0	0	n	$m*n$	m

eqn. type	y^2x	x^3	y^3
size	$n*m(m-1)/2$	$n(n-1)(n-2)/6$	$m(m-1)(m-2)/6$
+if $p > 2$	$n(n-1)$	$n*m$	$m(m-1)$
+if $p > 3$	n	0	m

eqn. type	x^3y	x^2y^2
size	$m*n(n-1)(n-2)/6$	$n(n-1)m(m-1)/4$
+if $p > 2$	$m(n-1)$	$m*n$
+if $p > 3$	$m*n$	0

La taille d'une union est simplement la somme de tailles des composantes. Par exemple si $p = 2$ on obtient la formule suivante utile dans 15.1.3 :

$$size_{XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2} = \frac{7}{12}nm + \frac{1}{4}(nm^2 - n^2m + n^2m^2) + \frac{1}{6}n^3m + n + m + 1.$$

13.3.5 Propriétés d'équations

Avant de décrire les principales attaques équationnelles, notons que ces 6 attaques, qui se généralisent les unes des autres et qui utilisent des dizaines de types d'équations, ont les propriétés suivantes qui nous ont amené à faire beaucoup de simulations :

1. On n'arrive à prévoir leur résultat que dans les cas simples.
2. On en trouve d'habitude plus que prévu.
3. Les attaques se comportent pratiquement de la même façon quelque soit n à partir d'un certain seuil n_0.
4. Ce seuil n_0 peut demander une quantité de mémoire considérable (p.ex. 840 Mo) pour vérifier ce qui se passe au-delà de ce seuil.
5. Pour HFE les propriétés des équations dépendent très nettement de $\log_q d$.

13.3.6 Cryptanalyse de C^*

Cette attaque surprenante avait été trouvée par Jacques Patarin pour le cryptosystème des japonais Matsumoto et Imai, connu aussi sous le nom de C^*. Ce cryptosystème peut être vu comme une version simplifiée de HFE avec le polynôme caché f qui est un monôme :

$$f(a) = a^{1+q^\alpha}$$

L'attaque de C^* avait été publiée dans [56]. Son principe est le suivant :

- Pour des raisons algébriques, voir [56], il existe des équations de type XY vraies avec probabilité 1 qui relient les couples clair-chiffré.
- L'existence de ces équations est invariable par les changement de variables S et T
- On peut détecter et calculer de telles équations qui existent sur la clef publique g.
- Dans ces équations de type XY, on substitue les valeur des y_i par les bits du chiffré que l'on cherche à déchiffrer.
- On obtient des équations linéaires sur les x_i.

13.3.7 Attaque de Multiples affines

Cette attaque proposée par J.Patarin dans [65] généralise la précédente mais considère en général des équations de types XY^2, XY^3, XY^4, qui deviennent linéaires quand on fixe la valeur de y.

13.3.8 Attaque de multiples de degré élevé - équations implicites

Ensuite Courtois et Patarin ont proposé d'utiliser des multiples de degré plus élevé, aussi appelés des équations implicites. Dans certains cas on arrive ainsi à casser une fonction.

Si par exemple les équations deviennent quadratiques après la substitution, et si leur nombre est de l'ordre de εn^2 on va pouvoir continuer l'attaque par linéarisation ou XL décrits dans le chapitre 7.

13.3.9 Attaque IXL

Cette attaque proposée par Courtois est surprenante et on peut prouver qu'elle contient la précédente.

D'abord on changera de notation en utilisant l à la place de y, la différence étant la même que dans le chapitre 7 :

Si on cherche à trouver un x tel que

$$y = g(x) = c$$

alors on posera

$$l_i = y_i(x_1, \ldots, x_n) - c_i.$$

Ainsi le problème de résoudre ces équations s'écrira sous une forme plus élégante et qui simplifiera bien des choses comme :

$$l = 0.$$

Jusque là, les types d'équations considérés ont été invariants par des changements de variables S et T qui apparaissent dans tant de schémas à clef publique multivariables.

Or, supposons qu'il existe sur une fonction g environ $n^2/2$ équations de type $x^2 l \cup X^3$. Cela veut dire que $n^3/3!$ termes de type X^3 peuvent être écrits comme des équations de type $x^2 l$. Parmi ces équations on peut éliminer $n^3/3! - n$ équations et autant de variables, ce qui donnera n équations de type $x^2 l \cup x$.

Ce type d'équations n'est plus invariant par S. Pourtant leur existence est prouvée, leur utilisation plus simple. On n'a plus guère besoin d'appliquer XL à la fin, elles deviennent linéaires déjà après la substitution de 0 à la place des l_i.

La pratique montre qu'au-delà de la raison théorique évoquée ci-dessus, dans certains cas de telles équations existent aussi sans cette raison. Elles sont un outil de cryptanalyse assez expérimental qui fonctionne très bien comme le montrent les nombreuses simulations qui suivent.

Chapitre 14

Simulations sur HFE

Les types d'équations ont été introduits dans 13.3.3.

14.1 Conventions

L'origine du nombre d'équations affiché dans les simulations qui suivent est volontairement déplacée d'un certain nombre afin de rendre les résultats plus clairs.

Nous allons écrire le nombre d'équations trouvé comme

$$\boxed{\mathbf{A} \rightarrow \mathbf{B}}$$

avec :

A est le nombre d'équations moins la prévision théorique calculée dans le chapitre suivant.

B est le nombre d'équations après la substitution par un y aléatoire, toujours moins la prévision théorique (souvent nulle pour B).

Le but de ces prévisions théoriques données ci-dessous, est de pouvoir repérer facilement si une attaque peut marcher ou pas. En effet, pour différentes raisons il y a toujours des équations, mais grâce à nos déplacements d'origine, pour une fonction MQ aléatoire (sans trappe) on s'attend à obtenir :

$$\boxed{\mathbf{0} \rightarrow 0}$$

Signification : si le nombre relatif d'équations $B > 0$, c'est une faiblesse qui peut être exploitée par des attaques.

14.1.1 Les syzygies

Les résultats équationnels (A) pour HFE ont comme référence une prévision théorique d'un certain nombre d'équations qui existent toujours pour (presque) tout système d'équations MQ.

Ces équations sont appelées syzygies dans le langage de bases de Gröbner. Leur nombre sera calculé ici par des raisonnements de combinatoire élémentaire.

14.1.2 Les calculs de l'origine de A

Pour tout type d'équations *type*, nous expliquons dans le tableau qui suit qu'un certain nombre d'équations existe toujours. Nous notons $trivial_{type}$ ce nombre.

Leur provenance est expliquée dans la colonne de droite. Pour cela on distingue les termes y_i et leurs expressions en tant que $y_i = \sum \sim x_j x_k \ldots$ notées $[y_i]$.

On pose $\delta_{p,2} = 1$ si $p = 2$, et $\delta_{p,2} = 0$ sinon.

type	$trivial_{\text{type}}$	their origin
$X \cup Y$	0	$\emptyset$
XY	0	$\emptyset$
$XY \cup X^2$	m	$y_i = [y_i]$
X^2Y	$\frac{1}{2}m(m+1) + \delta_{p,2}m$	the above and $y_i[y_j]=[y_i]y_j$ and if $\delta_{p,2}$: $y_i[y_i]=y_i$
$X^2Y \cup XY^2$	$m^2 + \delta_{p,2}m$	$y_i=[y_i]$ and $y_i[y_j]=y_i y_j$ and if $\delta_{p,2}$: $y_i[y_i]=y_i$
$X^2Y \cup XY^2 \cup Y^3$	$m^2 + \delta_{p,2}m$	the same as above.
$X^2Y \cup XY^2 \cup X^3$	$m^2 + \delta_{p,2}m + n * m$	the above and $x_i[y_j] = x_i y_j$.
$XY \cup x^2y$	$\frac{1}{2}m(m-1) + \delta_{p,2}m$	$y_i[y_j]=[y_i]y_j$ and if $\delta_{p,2}$: $y_i[y_i]=y_i$
$XY \cup x^2y \cup xy^2$	$\frac{1}{2}m(m-1) + \delta_{p,2}m$	the same as above.
$XY \cup x^2y \cup xy^2 \cup x^3y$	$nm^2 + \frac{1}{2}m(m-1) + \delta_{p,2}m$	the above and $x_i[y_j]y_k = x_i y_j y_k$.
$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$	$nm^2 + \frac{1}{2}m(m-1) + \delta_{p,2}m + \frac{1}{3}m(m-1)(m-2) - \frac{1}{2}m(m-1)$ +640(21)?	the above and $y_i[y_j]y_k=[y_i]y_j y_k$ eliminate the terms $\{y^2\}$.

14.1.3 Les calculs de l'origine de B

Le nombre choisi pour origine de B est égal à 0 quand le type d'équations n'inclut pas les équations quadratiques en les x_i et n sinon. Des cas plus complexes n'apparaissent jamais dans le présent mémoire.

14.2 Les prévisions et MQ

Ces simulations ont été faites pour MQ, ce qui correspond à HFE avec $d \to \infty$, le cas limite de HFE.

$m = 22, d = \infty$

		Equation type							
n	Mb.	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$XY^2 \cup X^3$
11	8	44→11	44→11	77→44	704→44	2186→11	2945→44	4485→44	2897→33
12	9	0→0	0→0	44→44	748→56	2047→12	3059→56	4599→56	2979→12
13	11	0→0	0→0	1→1	748→69	1176→13	2464→69	2464→69	2464→69
14	12	0→0	0→0	0→0	692→83	337→14	1924→83	1924→83	1980→139
15	14	0→0	0→0	0→0	567→98	0→0	1029→98	1029→98	1154→223
16	16	0→0	0→0	0→0	359→114	0→0	359→114	359→114	567→322
17	19	0→0	0→0	0→0	53→53	0→0	53→53	53→53	359→359
18	22	0→0	0→0	0→0	0→0	0→0	0→0	0→0	53→53
19	25	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

Ces résultats montrent que pour n suffisamment grand, MQ se comporte conformément aux prévisions théoriques du chapitre précédent.On peut également les voir comme une façon de montrer que l'attaque XL marche jusqu'à un certain n.

Maintenant nous allons faire les mêmes tableaux pour des MQ avec trapdoor, qui sont des véritables HFE. Cela montrera clairement les faiblesses de HFE.

14.3 Des attaques de HFE sur $\mathbb{F}_2$

Ce premier tableau à lui seul contient toutes les attaques équationnelles intéressantes de point de vue pratique pour la cryptanalyse de HFE sur $\mathbb{F}_2$.

n=21	Equation type					
d	XY	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y	$X^2Y \cup XY^2 \cup X^3$	$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$
3	42→19	693→19	1995→19	882→210	2688→484	→
4	21→21	441→21	1995→21	630→210	2688→484	→
5	1→1	232→18	1177→18	357→144	1806→484	→
8	1→1	170→20	1094→20	336→184	1764→484	→
9	0→0	126→18	672→18	231→124	1134→337	→
16	0→0	43→20	568→20	168→144	1092→379	→
17	0→0	0→0	63→16	84→84	357→169	→
24	0→0	0→0	22→18	84→84	315→311	→
32	0→0	0→0	0→0	64→64	315→315	→
33	0→0	0→0	0→0	0→0	147→147	→
64	0→0	0→0	0→0	0→0	147→147	4739→20
65	0→0	0→0	0→0	0→0	42→42	1911→17
96	0→0	0→0	0→0	0→0	42→42	1638→21
128	0→0	0→0	0→0	0→0	42→42	1547→20
129	0→0	0→0	0→0	0→0	0→0	0→0

Des résultats qui suivent montreront que ce comportement ne change pas quand n varie ainsi que d'autres types d'équations et plusieurs autres cas.

Certains calculs n'ont pas pu être faits. La plupart du temps c'était parce que la mémoire nécessaire dépasse 1 Go, où alors que le résultat se déduit clairement par rapport aux autres.

n=17

d	Equation type							
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y	$X^2Y \cup XY^2 \cup X^3$	$XY \cup x^2y \cup xy^2 \cup x^3y$	$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$
3	34→16	51→32	459→15	1309→15	578→134	1700→256	→	→
4	17→16	51→48	323→17	1309→17	442→133	1700→256	→	→
5	1→1	34→34	188→15	817→15	289→115	1190→256	→	→
8	1→1	34→34	153→16	765→16	272→136	1156→256	→	→
9	0→0	17→17	102→14	476→14	187→101	782→205	→	→
16	0→0	17→17	35→14	392→14	136→111	748→239	2k→	→
17	0→0	0→0	0→0	51→17	68→68	289→252	2k→	→
24	0→0	0→0	0→0	26→21	68→68	255→251	→	→
32	0→0	0→0	0→0	0→0	52→52	255→255	2k→16	→
33	0→0	0→0	0→0	0→0	0→0	119→119	2k→17	→
64	0→0	0→0	0→0	0→0	0→0	119→119	2k→16	→
65	0→0	0→0	0→0	0→0	0→0	34→34	→	→
96	0→0	0→0	0→0	0→0	0→0	34→34	→	→
128	0→0	0→0	0→0	0→0	0→0	34→34	→	→
129	0→0	0→0	0→0	0→0	0→0	0→0	→	→

n=21

d	Equation type							
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y	$X^2Y \cup XY^2 \cup X^3$	$XY \cup x^2y \cup xy^2 \cup x^3y$	$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$
3	42→19	63→41	693→19	1995→19	882→210	2688→484	→	→
4	21→21	63→62	441→21	1995→21	630→210	2688→484	→	→
8	1→1	42→42	170→20	1094→20	336→184	1764→484	→	→
9	0→0	21→21	126→18	672→18	231→124	1134→337	→	→
16	0→0	21→21	43→20	568→20	168→144	1092→379	→	→
17	0→0	0→0	0→0	63→16	84→84	357→169	→	→
32	0→0	0→0	0→0	0→0	64→64	315→315	→	→
33	0→0	0→0	0→0	0→0	0→0	147→147	→	→
64	0→0	0→0	0→0	0→0	0→0	147→147	602→20	4379→20
65	0→0	0→0	0→0	0→0	0→0	42→42	147→17	2551→17
96	0→0	0→0	0→0	0→0	0→0	42→42	2→2	1628→19
128	0→0	0→0	0→0	0→0	0→0	42→42	0→0	1547→20
129	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

n=22

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
3	44→22	44→22	66→44	968→231	990→22	2904→231	2860→231	3388→803
4	22→22	44→22	66→64	682→231	572→22	2904→231	2860→231	3388→803
5	1→1	1→1	44→44	374→152	133→20	1914→151	2002→151	2024→723
8	1→1	1→1	44→44	352→194	89→21	1364→193	1408→193	1980→766
9	0→0	0→0	22→22	242→129	0→0	836→130	836→130	1232→525
16	0→0	0→0	22→22	176→152	0→0	748→151	748→151	1188→591
17	0→0	0→0	0→0	88→88	0→0	176→131	176→131	374→329
32	0→0	0→0	0→0	67→67	0→0	110→110	110→110	330→330
33	0→0	0→0	0→0	0→0	0→0	0→0	0→0	154→154
64	0→0	0→0	0→0	0→0	0→0	0→0	0→0	154→154
65	0→0	0→0	0→0	0→0	0→0	0→0	0→0	44→44
128	0→0	0→0	0→0	0→0	0→0	0→0	0→0	44→44
129	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

n=22

d	Equation type							
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y	$X^2Y \cup XY^2 \cup X^3$	$XY \cup x^2y \cup xy^2 \cup x^3y$	$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$
3	44→22	66→44	759→22	2189→22	986→231	→803	→	→
4	22→22	66→44	473→22	2189→22	682→231	→803	→	→
5	1→1	44→44	243→20	1277→20	374→152	2024→723	→	→
8	1→1	44→44	180→22	1192→22	352→194	1980→766	→	→
9	0→0	22→22	132→19	726→19	242→129	1232→525	→	→
16	0→0	22→22	45→21	617→21	176→152	1188→591	3→21	→
17	0→0	0→0	0→0	66→22	88→88	374→329	5k→22	→
24	0→0	0→0	0→0	23→18	→	→	5k→20	→
32	0→0	0→0	0→0	0→0	67→67	330→330	5k→21	→
33	0→0	0→0	0→0	0→0	0→0	154→154	→	→
64	0→0	0→0	0→0	0→0	0→0	154→154	→	→
65	0→0	0→0	0→0	0→0	0→0	44→44	→	→
96	0→0	0→0	0→0	0→0	0→0	→	→	→
128	0→0	0→0	0→0	0→0	0→0	44→44	0→0	→
129	0→0	0→0	0→0	0→0	0→0	0→0	? ? ?→0	→

n=25

d	Equation type							
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y	$X^2Y \cup XY^2 \cup X^3$	$XY \cup x^2y \cup xy^2 \cup x^3y$	$XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$
3	50→25	75→50	975→25	2825→25	→	→	→	→
8	1→1	50→50	202→23	1502→23	→	→	→	→
9	0→0	25→25	150→22	922→22	→	→	→	→
16	0→0	25→25	51→24	776→24	→	→	→	→
17	0→0	0→0	0→0	75→25	100→100	→	→	→
32	0→0	0→0	0→0	0→0	76→76	→	→	→
65	0→0	0→0	0→0	0→0	0→0	→	→	→
96	0→0	0→0	0→0	0→0	0→0	→	¿2→	→
129	0→0	0→0	0→0	0→0	0→0	→	0→0	→

14.4 Comparaison de HFE sur $\mathbb{F}_2$, $\mathbb{F}_4$ et $\mathbb{F}_{16}$

Il s'agit ici de comparer HFE avec $q = 2, 4, 16$. Il en résultera que HFE sur $\mathbb{F}_2$ est le plus solide.

Les HFE quadratiques sur $\mathbb{F}_2$.

Remarque : certains tableaux ont été faits avec le même n mais ne sont pas identiques, car ils contiennent d'autres types d'équations.

n=17

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
3	34→16	34→16	51→32	578→134	595→16	1428→136	1666→136	1938→408
8	1→1	1→1	34→34	272→136	69→16	884→136	918→136	1190→408
9	0→0	0→0	17→17	187→101	0→0	561→100	561→100	782→321
16	0→0	0→0	17→17	136→111	0→0	493→115	493→115	748→370
17	0→0	0→0	0→0	68→68	0→0	136→99	136→99	289→252
32	0→0	0→0	0→0	52→52	0→0	85→85	85→85	255→255
33	0→0	0→0	0→0	0→0	0→0	0→0	0→0	119→119
128	0→0	0→0	0→0	0→0	0→0	0→0	0→0	34→34
129	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

Les HFE quadratiques sur $\mathbb{F}_4$.

n=17

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
16	0→0	0→0	51→51	323→136	0→0	1156→136	1207→136	1598→527
64	0→0	0→0	34→34	153→119	0→0	697→119	967→119	1088→510
65	0→0	0→0	17→17	17→17	0→0	306→17	306→17	697→408
256	0→0	0→0	17→17	17→17	0→0	306→17	306→17	680→391
257	0→0	0→0	0→0	0→0	0→0	0→0	0→0	221→221
1024	0→0	0→0	0→0	0→0	0→0	0→0	0→0	204→204
1025	0→0	0→0	0→0	0→0	0→0	0→0	0→0	119→119
4096	0→0	0→0	0→0	0→0	0→0	0→0	0→0	119→119
16384	0→0	0→0	0→0	0→0	0→0	0→0	0→0	34→34
16385	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

n=22

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
16	0→0	0→0	66→66	374→219	0→0	1782→219	1848→219	2728→677
17	0→0	0→0	44→44	176→132	0→0	1100→132	1100→132	1870→902
64	0→0	0→0	44→44	110→66	0→0	1034→66	1034→66	1848→880
65	0→0	0→0	22→22	22→22	0→0	506→22	506→22	1144→660
256	0→0	0→0	22→22	22→22	0→0	506→22	506→22	1144→660
257	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
1024	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
1025	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
4096	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
16384	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
16385	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
65536	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
65537	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
1048577	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418

n=25

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
16	0→0	0→0	75→75	425→246	0→0	2250→246	2325→246	3550→1470
256	0→0	0→0	25→25	25→25	0→0	650→25	650→25	1400→775
257	0→0	0→0	0→0	0→0	0→0	0→0	0→0	325→325
4096	0→0	0→0	0→0	0→0	0→0	0→0	0→0	175→175
16384	0→0	0→0	0→0	0→0	0→0	0→0	0→0	50→50
16385	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

Les HFE quadratiques sur $\mathbb{F}_{16}$.

n=22

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
32	0→0	0→0	66→66	286→132	0→0	1694→132	1694→132	2684→748
65536	0→0	0→0	22→22	22→22	0→0	506→22	506→22	1144→660
65537	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418
1048577	0→0	0→0	0→0	0→0	0→0	0→0	0→0	418→418

n=25

d	Equation type							
	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
32	0→0	0→0	75→75	326→151	0→0	2151→151	2151→151	3500→1475
65536	0→0	0→0	25→25	25→25	0→0	650→25	650→25	1400→775
65537	0→0	0→0	0→0	0→0	0→0	0→0	0→0	325→325
1048577	0→0	0→0	0→0	0→0	0→0	0→0	0→0	175→175

14.5 Comparaisons d'attaques avec n croissant

Les but de ces tableaux est de montrer que quand n croît, le nombre d'équations trouvés décroît pour les valeurs initiales n petit, et ensuite se stabilise sur un nombre qui dépende de m, par exemple $2m$.

$m = 22, d = 128$

n	Mb.	Equation type							
		XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
14	12	0→0	0→0	0→0	692→83	337→14	1924→83	1924→83	1980→139
16	16	0→0	0→0	0→0	359→114	0→0	359→114	359→114	567→322
18	22	0→0	0→0	0→0	0→0	0→0	0→0	0→0	53→53
19	25	0→0	0→0	0→0	0→0	0→0	0→0	0→0	44→44
22	37	0→0	0→0	0→0	0→0	0→0	0→0	0→0	44→44

$m = 34, d = \infty$

n	Mb.	Equation type							
		XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
17	79	0→0	0→0	0→0	1547→119	1326→17	4896→119	4896→119	4998→221
19	98	0→0	0→0	0→0	1020→156	0→0	1020→156	1020→156	1343→479
21	123	0→0	0→0	0→0	0→0	0→0	0→0	0→0	560→560
22	137	0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0
27		0→0	0→0	0→0	0→0	0→0	0→0	0→0	0→0

$m = 34, d = 128$

n	Mb.	Equation type							
		XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
17	79	0→0	0→0	0→0	1547→119	1326→17	4896→119	4896→119	4998→221
20	109	0→0	0→0	0→0	560→176	0→0	560→176	560→176	1020→636
22	137	0→0	0→0	0→0	0→0	0→0	0→0	0→0	68→68
27		0→0	0→0	0→0	0→0	0→0	0→0	0→0	68→68

Ce tableau est fait avec $m = 34$, $d = 2^{24} + 1$ et sur $\mathbb{F}_{16}$

n	Mb.	Equation type							
		XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup Y^3$	$X^2Y \cup XY^2 \cup Y^3 \cup X^3$
17	78	0→0	0→0	0→0	1547→119	1326→17	4896→119	4896→119	4998→221
19	97	0→0	0→0	0→0	1020→156	0→0	1020→156	1020→156	1343→479
21	120	0→0	0→0	0→0	0→0	0→0	0→0	0→0	560→560
22	133	0→0	0→0	0→0	0→0	0→0	0→0	0→0	68→68
23	151	0→0	0→0	0→0	0→0	0→0	0→0	0→0	68→68

Et encore quelque résultats divers avec le degré 128 sur $\mathbb{F}_2$.

$d = 128$

m,n	Mb.	Equation type							
		$X \cup Y$	XY	$XY \cup Y^2$	$XY \cup X^2$	X^2Y	XY^2	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup X^3$
44,19	120	0→0	0→0	→	0→0	2525→146	→	8871→146	9004→279
44,21	160	0→0	0→0	→	0→0	1859→187	→	1859→187	2265→593
44,23	200	0→0	0→0	→	0→0	528→232	→	528→232	1287→991
44,25	240	0→0	0→0	→	0→0	0→0	→	0→0	88→88
44,26	284	0→0	0→0	→	0→0	0→0	→	0→0	88→88
80,27		0→0	0→0	→	0→0	6525→298	→	→	→
80,29	180	0→0	0→0	→	0→0	4155→	→	→	→
80,31	233	0→0	0→0	→	0→0	480→416	→	→	→
80,32	260	0→0	0→0	→	0→0	0→0	→	→	→
93,33	390	0→0	0→0	→	0→0	1426→468	→	→	→
93,36	540	0→0	0→0	→	0→0	0→0	→	→	→

14.6 Attaques itérés

Le principe des attaques itérés :

1. Au début il y a m équations avec n variables.
2. La première itération doit trouver des nouvelles équations non-triviales sur les variables.
3. Cela donne un nouveau $m' > m$.
4. On répète jusqu'à obtenir assez d'équations linéaires pour résoudre.

Attaques avec les degrés 16,32,64 sur$\mathbb{F}_2$.

$n = 22, d = 16$

m	Mb.	Equation type					
		$X \cup Y$	XY	$XY \cup X^2$	X^2Y	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup X^3$
22− > 44	.3	0→0	0→0	22→22	→	→	→
44− > 110	.6	0→0	24→2	88→66	→	→	→
110− > 240	2	2→2	869→22	990→143	→	→	→
240		22→22	3762→22	3753→13	→	→	→

$n = 22, d = 32$

m	Mb.	Equation type					
		$X \cup Y$	XY	$XY \cup X^2$	X^2Y	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup X^3$
22− > 89	7	0→0	0→0	0→0	67→67	→	→
89− > 240	80	0→0	276→19	418→161	6957→161	→	→
240		19→19	3759→19	3750→10	→	→	→

Attaques avec le degré 128 sur $\mathbb{F}_2$.

$n = 22, d = 128$

m	Mb.	Equation type					
		$X \cup Y$	XY	$XY \cup X^2$	X^2Y	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup X^3$
17− > 51	7	0→0	0→0	0→0	0→0	0→0	34→34
51− > 51	146	0→0	85→0	102→17	1887→17	16541→17	16892→368
51− > 51		0→0	85→0	102→17	1887→17	16541→17	16892→368
51− > 51		0→0	85→0	102→17	1887→17	16541→17	16892→368
17− > 153	7	0→0	0→0	0→0	0→0	0→0	34→34
153− > 152	90	0→0	442→0	459→17	14811→152	→	→
152		16→16	1920→16	2056→152	20348→152	→	→
17− > 170	7	0→0	0→0	0→0	0→0	0→0	34→34
170− > 153	90	0→0	556→0	573→17	17394→153	→	→
153	90	17→17	1938→17	2074→153	20502→153	→	→

$n = 22, d = 128$

m	Mb.	Equation type					
		$X \cup Y$	XY	$XY \cup X^2$	X^2Y	$X^2Y \cup XY^2$	$X^2Y \cup XY^2 \cup X^3$
22− > 44	28	0→0	0→0	0→0	0→0	0→0	44→44
44− > 44	176	0→0	44→0	66→22	1496→22	13552→22	14036→506
22− > 198	28	0→0	0→0	0→0	0→0	0→0	44→44
198− > 22	377	0→0	968→0	990→22	21131→22	→	→
22− > 198	28	0→0	0→0	22→0	275→0	506→0	1012→506

Chapitre 15

Application des attaques équationnelles

Nous allons rappeler le principe des attaques équationnelles, introduit dans 13.3.2 et essayer d'optimiser l'utilisation qui est faite du fait que ce type d'attaques existent et fonctionnent, comme le montrent les simulations de 14.

Pour HFE la plus efficace de toutes ces attaques s'avère être l'attaque IXL décrite dans 13.3.9. Rappelons que cette attaque consiste à :

1. Translater le problème de sorte que l'instance à résoudre soit $l = 0$ au lieu de $y = cste$.
2. Sachant (par simulations analogues avec petite taille) qu'il existe un certain nombre d'équations d'un certain type, par example $x^2l \cup X^3$ selon les conventions de 13.3.3, écrire les équations avec une nouvelle variable pour chaque coefficient d'un terme présent dans ces équations, par example le terme $x_1x_2l_5$ est bien dans $x^2l \cup X^3$.
3. Avec un nombre suffisant de couples (clair, chiffré), on arrive à calculer tous les coefficients présents dans ces équations.
4. On substitue $l = 0$ et grâce à leur forme spécifique exigée dans 13.3.9, il ne reste que des équations linéaires en x_i.

Dans cette partie nous allons montrer plusieurs façons d'exploiter ce type d'équations dans les attaques réalistes.

15.1 Interprétation des attaques expérimentales

Dans toutes les simulations de 14 on s'aperçoit que quand d décroît, les équations disparaissent et on a besoin d'équations de plus en plus complexes. A partir d'un certain n (phénomène de seuil) on observe une très grande régularité dans les simulations. Le nombre d'équations semble s'exprimer comme un polynôme de petit degré en n, par exemple $2n$, et l'existence ou non des équations ne change pas avec n.

On peut observer que les HFE de degrés $d = q^k + 1..q^{k+1}$ se comportent toujours de façon analogue. Ceci est en accord parfait avec l'attaque de Shamir-Kipnis de 13.2.5) pour laquelle ce genre de comportement est prouvé. Vu nos simulations de 14, et un lien visible (mais pas prouvé) avec l'attaque de Shamir-Kipnis il paraît plus toute à fait fondé de prétendre que :

Conjecture 15.1.0.1 *Le problème HFE de degré d admet les équations de type* $X^{\frac{1}{2}\lceil \log_q d \rceil - 1} Y$.

Quelles sont les implications de cette conjecture ?

15.1.1 La complexité de l'attaque de Courtois

La taille des équations de la conjecture sera de l'ordre de

$$size = n^{(1/2 \log_q d)}.$$

Ainsi, si l'on suppose que la réduction de Gauss est cubique, la complexité de l'attaque équationnelle est environ

$$WF = n^{(3/2 \log_q d)}.$$

15.1.2 Des attaques réalistes de HFE pour $d \leq 24$.

Nos simulations montrent que HFE avec le degré $d <= 24$ admet des équations de type $XY + x^2y + xy^2$.

On en obtient entre $\mathcal{O}(n)$ et $\mathcal{O}(n^2)$, ce qui est suffisant pour casser, car ces équations deviennent linéaires après la substitution de $y = 0$ (attaque IXL, voir 13.3.9)

Exemple d'application pour un HFE de $n = 64$ bits, $d = 24$. La taille de nos équations est :

$$size_{XY \cup x^2y \cup xy^2}(64, 64) = \mathcal{O}(n^3)$$

Le calcul exact en utilisant les tableaux de 13.3.4 donne $size = 262\ 273$.

La mémoire requise par l'attaque est $size^2/8 = \mathcal{O}(n^6) = 8$ Go.

Le temps de calcul par la réduction de Gauss cubique standard, et en tenant compte du fait qu'un processeur peut additionner 64 bits à la fois grâce à un XOR, sera de l'ordre de $size^3/64 \approx 2^{48}$ cycles d'horloge.

Il n'y a pas de doute que HFE (dans la version algébrique de base) n'est pas solide pour $d \leq 24$. La complexité de notre attaque est au plus n^9.

15.1.3 Attaque directe pour le Challenge 1

Ceci est la seule attaque connue meilleure que la recherche exhaustive, pour le HFE Challenge 1 décrit dans 13.1. Un exemple concret de clef publique pour le Challenge 1 peut être téléchargé sur la page internet de HFE [70]. Un peu plus bas, dans 15.2 nous décrirons des améliorations de cette même attaque.

Dans ce challenge on a $d = 96$, l'attaque que nous décrivons ici marchera jusqu'à $d = 128$. Dans les tableaux de nos simulations, on voit dans 14.3 que HFE de degré 96 admet un grand nombre d'équations de type $1 \cup x \cup y \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$. Il reste toujours environ n de telles équations après leur substitution avec $y = 0$ (principe de l'attaque IXL de 13.3.9). Leur taille est calculée dans 13.3.4 et vaut pour $m = n = 80$:

$$size_{XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2}(80, 80) = 17\ 070\ 561$$

La mémoire nécessaire pour récupérer ce type d'équations n'est pas très réaliste : $size^2/8 = 33$ Tera-octets. Le temps de l'attaque est environ $2 \cdot size^3/64 \approx 2^{62}$ opérations CPU.

15.2 Attaques avec réconciliation et distillation

Ces attaques ont été proposées par l'auteur et décrites brièvement dans [68]. Il s'agit des astuces techniques permettant de calculer les équations implicites par morceaux, en économisant le temps de calcul et la mémoire, avec une synchronisation qui permet de bien les 'recoller' car les équations ne sont jamais récupérées qu'à une combinaison linéaire près. Cela peut marcher jusqu'à un certain seuil que nous allons calculer.

15.2.1 Réduire la taille

Puisque la taille *size* des équations, est trop grande, il est intéressant de les calculer par morceaux. Au lieu de choisir un x aléatoire dans les couples x, y utilisés pour générer des lignes de la matrice qui sert à récupérer les équations, on choisit x dans un sous-espace affine comme suit :

1. Soit n_a la dimension $n_a \leq n$. On va prendre un sous espace de la forme $\{x \mid x_{n_a+1} = 0, \ldots, x_n = 0\}$
2. Cela réduit le nombre de coefficients qu'il est nécessaire de récupérer dans les équations. On ne va pas se soucier des coefficients qui contiennent un de $x_j, j = n_a + 1 \ldots n$.
3. Ainsi on obtient des équations plus petites en taille, vraies avec probabilité 1 sur un sous-espace.
4. Il est clair qu'elles contiennent les équations "entières" recherchées, modulo une combinaison linéaire, prises sans les termes contenant un de $x_j, j = n_a + 1 \ldots n$ qui sont nuls.
   Ces équations qui reflètent les équations "entières" sont appelés les "cast equations".
5. Malheureusement, si n_a est trop petit, d'autres équations vont apparaître qui sont appelées des équations artificielles "artificial", et qui s'ajoutent à ce qu'on trouve sans être des "cast".

On ajoute encore que évidemment :

Proposition 15.2.1.1 *Des "cast" équations des équations triviales, telles que définies dans 13.3.2, restent triviales, de même que les "cast" des équations artificielles ci-dessus restent artificielles.*

L'idée de l'attaque est de filtrer les équations artificielles, mais d'abord il faudrait savoir combien elles sont.

15.2.2 Les équations artificielles

Pour simplifier, on va se restreindre au cas binaire $q = 2$. Les équations appelés "artificielles" (notion informelle) sont dues à une dimension trop petite n_a, et le degré trop petit des expressions en y_i. On considère tous les termes, par example les $x_i x_j y_k$, en tant que les polynômes de degré D en les $x_i, i = 1..n_a$, avec D qui va dépendre du type d'équation. Le nombre maximum de tels termes est pour les $x_i \in \mathbb{F}_2$:

$$Dim_{\mathbb{F}_2}(n_a, D) \stackrel{def}{=} \sum_{i=0}^{D} \binom{n_a}{i}$$

Quand n_a et D sont suffisamment petits, la valeur de $Dim_{\mathbb{F}_2}(n_a, D)$ s'avère plus petite que le nombre de tous les termes coefficients $size(n_a, n_b)$.

Les $x_i x_j y_k$, quand ils sont réécrit comme des combinaisons linéaires des $x_i x_j x_k x_l$ se retrouvent dans un espace de dimension trop petite. Soit $artificial(n_a, n_b)$ le nombre d'équations artificielles obtenues. Il contient malheureusement aussi des équations qui existent pour d'autres raisons, à condition qu'elles soit moins nombreuses que la différence de dimension. On s'attend à ce que :

Conjecture 15.2.2.1

$$artificial(n_a, n_b) \geq size(n_a, n_b) - Dim(n_a, D)$$

S'il n'y pas de raison algébriques ou combinatoires supplémentaires, pour qu'il existe d'avantage d'équations, on s'attend à voir (à peu près) l'égalité en pratique. Ce ne sera pas le cas si le nombre d'équations existantes pour des raisons algébriques (équations non triviales) est supérieur.

Ce ne sera pas le cas non plus, si pour un sous ensemble strict du type d'équation considéré, le degré de $D_{subtype}$ diminue et fait que $Dim(n_a, D_{subtype})$ décroît plus vite que $size_{subtype}(n_a, n_b)$. A ce moment là il est certain que l'on aura d'avantage d'équations. Cela arrive rarement, mais sûrement, par example :

$Dim(22, D_{XY \cup x^2y \cup xy^2 \cup x^3y}) - size_{XY \cup x^2y \cup xy^2 \cup x^3y}(22, 22) = 9130$, et

$Dim(22, D_{XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2}) - size_{XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2}(22, 22) = -12122$.

On peut s'attendre à ce que :

Conjecture 15.2.2.2

$$artificial_{type}(n_a, n_b) = \max_{t \subset type} Dim(n_a, D_t) - size_t(n_a, n_b)$$

Il n'est donc pas trivial du tout de comprendre exactement ce qui se passe. En pratique nous avons vérifié la validité de la formule de la Conjecture 15.2.2.1 sur plusieurs examples et cela semble marcher. Par example

$$\begin{aligned}\text{non-trivial}_{X^2Y}(15, 22) &= \text{artificial}_{X^2Y}(15, 22) - \text{trivial}_{X^2Y}(15, 22) = \\ &= \text{size}_{X^2Y}(15, 22) - \text{Dim}(15, 4) - \text{trivial}_{X^2Y}(15, 22) = 2783 - 1941 - 275 = \mathbf{567}\end{aligned}$$

En effet dans les 3 tables présentées on trouve 567 dans la ligne $n_a = 15$.

$n = 22, d = 16$

n_a	Equation type				
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y
15	0→0	22→22	484→15	1056→15	**567**→98
16	0→0	22→22	261→16	833→16	359→114
17	0→0	22→22	86→17	658→17	200→131
18-22	0→0	22→22	45→n_a	617→n_a	176→149

$n = 22, d = 24$

n_a	Equation type				
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y
15	0→0	0→0	484→15	946→15	**567**→98
16	0→0	0→0	261→16	327→16	359→114
17	0→0	0→0	0→0	40→17	88→88
18	0→0	0→0	0→0	25→18	88→88
19-22	0→0	0→0	0→0	23→n_a	88→88

$n = 22, d = \infty$

n_a	Equation type				
	XY	$XY \cup X^2$	$XY \cup x^2y$	$XY \cup x^2y \cup xy^2$	X^2Y
15	0→0	0→0	385→15	532→15	**567**→98
16	0→0	0→0	146→16	146→16	359→114
17	0→0	0→0	0→0	0→0	53→53
18	0→0	0→0	0→0	0→0	0→0
19-22	0→0	0→0	0→0	0→0	0→0

15.2.3 Cas limite des équations artificielles

Ce type d'équations existent toujours à partir d'un certain seuil, même si les y_i ne sont pas de petit degré, et même pour des fonctions aléatoires de degré maximum. La formule de la Conjecture 15.2.2.1 reste valable :

$$Dim_{\mathbb{F}_2}(n_a, \infty) = \sum_{i=0}^{\infty} \binom{n_a}{i} = 2^{n_a}.$$

$$artificial(n_a, n_b) \geq 2^{n_a} - size(n_a, n_b)$$

On peut aussi faire une démonstration directe : il y a 2^{n_a} valeurs possibles de x dans le sous-espace choisi. On considère un matrice de taille $2^{n_a} \times size$ avec des lignes qui correspondent à toutes les 2^{n_a} paires x, y possibles, et les colonnes correspondant à tous les termes possibles en $x_i y_j$ etc. Si $2^{n_a} < size$ il n'y a pas assez de x possibles, le rang de la matrice est au plus 2^{n_a} et on obtient au minimum $2^{n_a} - size$ équations qui relient des termes en $x_i y_j$ etc.

Exemple : Le problème est de récupérer une clef DES key k sachant que $E_k("i") = y_i, i = 1..2^{20}$. $k = (k_1 \dots k_{56}$ est sur 56 bits, $Y = \{y_i\}$ a 2^{26} bits = 8 Mo. On va considérer les équations de type KY^2 :

$$size_{KY^2} \approx 56 * \frac{(2^{26})^2}{2} \approx 2^{56.8} > 2^{56}$$

Ce qui montre qu'il y a environ 2^{55} équations de type KY^2, non pas pour de raisons algébriques, mais purement artificielles. Il est par contre difficile de les récupérer directement car cela nécessiterai au moins $(2^{56})^2 = 2^{132}$ de mémoire.

15.2.4 Le seuil de réconciliation

On suppose $n_b = n$ fixé. Le seuil de réconciliation appelé $n_{art}(type)$ sera défini comme le seuil d'apparition des équations artificielles

Reconcilliation Condition 15.2.4.1

$$artificial_{type}(n_a, n_b) > 0$$

Par example pour $n = n_b = 80$, nous avons calculé à l'aide de la Conjecture 15.2.2.1

$$n_{art}(XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2) = 38$$

Les équations artificielles brouillent complètement les "casts" des équations (non-triviales). On peut considérer que les équations artificielles sont les "cast" modulo un certain nombre d'équations "parasites" ajoutées.

L'idée de la réconciliation consiste à récupérer l'espace complet d'équations implicites recherchées dans l'attaques IXL, avec l'aide des plusieurs "cast" de ces équations sur plusieurs sous-espaces S_i couvrant tout l'espace $S = \mathbb{F}_2^n$ des x. Par définition, une équation qui est de type "cast" sur un sous-espace S_i contient uniquement les termes qui ne disparaissent pas sur tout S_i, par example parce que $X_3 = 0$ dans S_i. On note T_{S_i} cet ensemble de termes. La réconciliation serait évident si à l'origine il n'y avait qu'une équations à récupérer E_1. On va alors progressivement récupérer tous ses coefficients.

Dans le cas de plusieurs équations, $E_1, \dots, E_{eq}$, comme la récupération des "casts" se fait modulo une combinaison linéaire, on a besoin d'une série de sous espaces S_i tels que les intersections des ensembles de termes T_{S_i} $T_{S_i} \cap T_{S_{i+1}}$ soient suffisamment grandes de sorte à ce que la taille de l'ensemble de termes qui sont dans l'intersection soit assez grande pour déterminer une façon unique de les recomposer. On appelle eq_{S_i} le nombre d'équations trouvé dans chaque "cast".

Proposition 15.2.4.2 *Si*

$$card(T_{S_i} \cap T_{S_{i+1}}) \geq eq_{S_i} \leq eq_{S_{i+1}}$$

et en supposant que toutes les équations "cast" sur S_i *sont linéairement indépendantes quand on laisse que des termes de* $T_{S_i} \cap T_{S_{i+1}}$.

Alors la réconciliation peut être faite de façon unique.

C'est assez évident, les équations de S_{i+1} restreintes à $T_{S_i} \cap T_{S_{i+1}}$, ont assez de coefficients pour ne pas confondre leur combinaisons linéaires.

15.2.5 La distillation sur l'example de challenge 1.

Cette attaque est encore une astuce technique permettant de calculer les équations implicites au-delà du seuil de réconciliation. On prétend simplement qu'à fur et à mesure de la recomposition des équations on élimine toute sorte d'équations "parasites". On l'expliquera directement sur un example.

Le nombre d'équations artificielles calculé avec la Conjecture 15.2.2.1, est approximativement sous la forme

$$artificial(n_a, n_b) \approx \mathcal{O}(n_b^{C_1} + n_a^{C_2} n_b^{C_3} - n_a^{C_4})$$

avec $C_2 < C_4$. Cela explique pourquoi pour un n_b fixé, le nombre $artificial(n_a, n_b)$ croît d'abord pour n_a petit, puis décroît.

Soit n_a tel que :

Distillation Condition 15.2.5.1

$$artificial(n_a - 1, n_b) \geq artificial(n_a, n_b).$$

Il s'agit se placer juste après le maximum de $artificial(n_a, n_b)$.

Pour $n_b = 80$ et le type $XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$ la solution est $n_a \geq 30$. C'est plus petit que le seuil de réconciliation 38 obtenu dans 15.2.4.

La distillation fonctionne comme suit : on obtient les "cast" équations pour tous les $(n_b - n_a + 1)$ espaces S_i avec les indexes des x_i non-nuls étant limités respectivement à $[1..n_a], [2..n_a + 1], \ldots, [n_b - n_a + 1..n_b]$. On a lors :

Proposition 15.2.5.2 *La condition de distillation de 15.2.5.1 implique la condition de réconciliation 15.2.4.1*

En effet

$$card(T_{S_i} \cap T_{S_{i+1}}) = size(n_a - 1, n_b) \geq size(n_a - 1, n_b) - Dim_{\mathbb{F}_2}(n_a - 1, D) =$$
$$= artificial(n_a - 1, n_b) \geq artificial(n_a, n_b) = eq_{S_i} = eq_{S_{i+1}}.$$

La taille des équations que l'on va manipuler ne sera plus que :

$$size_{XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2}(30, 80) = 1\ 8315\ 11.$$

Donc on a plus besoin "que" de $size^2/8 = 390$ Go de mémoire au lieu de 33 Tb dans la version directe de cette attaque 15.1.3. On récupère les mêmes équations que dans 15.1.3 avec 87 fois moins de mémoire.

Le temps de calcul est $(80-30+1)*1.5*size^3/64 \approx 2^{62}$ opérations CPU, exactement le même qu'avant. L'attaque n'a pas été faite dans sa totalité, mais chacune des étapes avait été programmée et testée séparément pour s'assurer qu'elle fonctionne réellement.

Amélioration possible On peut gagner sans doute pas mal en améliorant le réduction de Gauss. Par example le problème de trouver (formellement) des équations de type $XY \cup x^2y \cup xy^2 \cup x^3y \cup x^2y^2$ ou similaire donne un système d'équations sparse. Il n'est pas évident si cette variante sera vraiment meilleure, dans une analyse faite dans [68] il semble que les équations ne seraient pas assez sparses pour en tirer vraiment profit.

15.2.6 La complexité asymptotique de la distillation

On va évaluer, très approximativement la plus puissante des versions améliorées de l'attaque sur HFE : la distillation. L'attaque marche, d'après 15.2.5.1, à partir d'un certain seuil n_a tel que la valeur de

$$artificial(n_a, n_b).$$

atteint un maximum. On calcule la dérivée, à la suite de 15.2.2.1 :

$$0 = (size(n_a, n_b) - Dim(n_a, D))'$$

ce qui donne approximativement pour le type $X^{\frac{1}{2}log_q d-1}L$ conjecturé dans 15.1.0.1 :

$$0 = \left((n_a)^{\frac{1}{2}log_q d-1} \cdot n_b - (n_a)^{\frac{1}{2}log_q d+1}\right)'$$

$$0 = \frac{1}{2}log_q(n_a)^{\frac{1}{2}log_q d-2} \cdot n_b - (n_a)^{\frac{1}{2}log_q d}$$

Cela donne environ $n_a \approx \sqrt{n_b} = \sqrt{n}$. On a donc

$$size \;\approx\; n\left(\sqrt{n}\right)^{\frac{1}{2}\log_q d} \;\approx\; n^{\frac{1}{4}\log_q d}.$$

La complexité asymptotique de l'attaque de distillation est donc environ :

$$n^{\frac{3}{4}\log_q d}.$$

Chapitre 16

La sécurité de HFE - résumé

16.1 La sécurité asymptotique

L'attaque de Shamir-Kipnis décrite dans 13.2.5 donne la complexité de

$$n^{\mathcal{O}(\log^2 d)}.$$

L'attaque Shamir-Kipnis-Courtois décrite dans 13.2.6 donne la complexité de

$$\left(\frac{ne}{\log_q d}\right)^{\omega(\log_q d+1)} \approx n^{3\log_q d}$$

Il en est de même pour l'attaque équationnelle décrite dans 15.1 qui semble donner des meilleurs résultats en pratique et qui avait été évaluée dans 15.1.3 :

$$n^{3/2\log_q d}.$$

La plus puissante des versions améliorées de l'attaque (distillation) donnerait d'après 15.2.6 environ :

$$n^{\frac{3}{4}\log_q d}.$$

Il faut manier ce résultat avec une grande précaution. On y a négligé de nombreux facteurs qui sont certes asymptotiquement négligeables mais pas en pratique. Sur notre example, il s'est avéré dans 15.2.5, que la complexité de l'attaque améliorée en $n^{3/4\log_q d}$, compte des facteurs négligés ici, est en fait tout aussi grande que pour l'attaque équationnelle ordinaire de 15.1 qui est en $n^{3/2\log_q d}$.

16.1.1 La sécurité de HFE, d fixé

Si d est fixé, chacune des attaques ci-dessus est polynomiale.

16.1.2 La sécurité de HFE en général

En général, d est polynomial en n (ce qui permet au méthodes univariables de résoudre HFE en clef secrète en temps polynomial, voir 12.2 et [65, 70]. Cela donne une complexité asymptotique de :

$$e^{\log^2 n}.$$

Toutes ces attaques contre le problème HFE sont donc sous-exponentielles. Il y a peu d'espoir que HFE soit polynomial.

16.2 La sécurité de HFE Challenge 1 et Quartz

Le HFE Challenge 1 est un pur problème algébrique HFE (i.e. un 'basic HFE'), décrit dans [65, 70]. Les paramètres sont $d = 96$ et $n = 80$. La recherche exhaustive est en 2^{80}. L'attaque de Shamir-Kipnis de 13.2.5 donne environ 2^{152}. L'attaque de Shamir-Kipnis-Courtois de 13.2.6 donne 2^{62} avec 33 Tera-octets d'espace disque nécessaire. L'attaque de distillation de 15.2.5 donne également 2^{62} mais avec seulement 390 Giga-octets de disque.

Il n'y a actuellement aucune attaque structurelle connue contre Quartz [66, 67]. Par contre il y a un sous ensemble strict de Quartz qui est un basic HFE, et même si sa cryptanalyse ne donne rien sur la cryptanalyse de Quartz, il est intéressant de l'étudier. Ainsi l'attaque de Shamir-Kipnis de 13.2.5 donnerait environ 2^{188}. L'attaque de Shamir-Kipnis-Courtois de 13.2.6 donnerait 2^{114} comme calculé dans le tableau 24.7. Une cryptanalyse directe par l'attaque de distillation, difficile à évaluer avec précision pour autre chose que le Challenge 1, donnerait probablement entre $2^{70} - 2^{110}$ avec beaucoup de mémoire. Tout cela, on le rappelle, sans jamais permettre de cryptanalyser Quartz tout entier.

16.3 Conclusions

16.3.1 L'irréductible Gaulois

Il s'est avéré que HFE est un cryptosystème dont la sécurité est reliée à 4 problèmes difficiles auxquels est consacrée la présente thèse.

Les réductions de HFE $\rightsquigarrow$ MinRank $\rightsquigarrow$ MQ de Shamir-Kipnis (13.2,[71]) n'ont pas donné les résultats espérés. Au contraire, on obtient des attaques beaucoup plus performantes quand on ne les utilise pas et quand l'on s'attaque directement au problème HFE comme dans 13.3.2.

Chapitre 17

Les applications de HFE

HFE est à la base un cryptosystème **algébrique** dont la sécurité repose sur des problèmes bien définis, pour lesquels il existe des attaques d'une certaine complexité généralement sous-exponentielle.

De petites modifications de HFE peuvent détruire la structure algébrique qui permet des attaques, et donner des variantes de HFE dont la sécurité ne peut être actuellement envisagée que sur le plan **combinatoire**, c'est à dire que la compréhension de la façon de s'y prendre pour les attaquer est aujourd'hui quasiment nulle. Elles augmentent l'entropie de la clef publique, en y ajoutant par example une partie parfaitement aléatoire. Elles ont également la faculté de rapprocher le système d'équations d'un ensemble d'équations parfaitement aléatoire MQ.

En même temps les paramètres de la fonction trappe : notamment la possibilité de chiffrer et déchiffrer se trouve dégradée, sans autant la rendre inutilisable en pratique. Les 4 modifications présentées ici vont respectivement, ajouter / enlever, les équations publiques / ou les variables. Cela modifiera l'équilibre entre la taille de blocs à l'entrée et la sortie et influera sur la possibilité d'utiliser la fonction en signature ou en chiffrement, mais pas de façon définitive, car on peut combiner les modifications entre elles, voir 17.1.1.

17.1 De versions améliorées de HFE

Des variantes modifiées de base sont (avec les notations reprises de [7]) :

♣ **HFE–** C'est un HFE où l'on enlève un certain nombre r d'équations publiques [1]. L'idée d'enlever les équations afin de renforcer un cryptosystème multivariable avait été proposée pour la première fois par Shamir dans [75]. Une variante simplifiée de ce schéma est Flash [61, 62] proposé au standard Européen Nessie.[2]

♣ **HFEv** Il s'agit d'un HFE auquel on ajoute de nouvelles variables, appelées des variables de **v**inaigre.

[1] Dans l'article [76] écrit avec Patarin et Goubin nous montrons une attaque qui amène à suggérer que si le HFE est faible, il faut avoir au moins $q^r > 2^{64}$.

[2] Flash est en fait un $C^* - -$. Le "–" signifies qu'on enlevé beaucoup d'équations, et C^* peut être vu comme une variante simplifiée de HFE comme nous l'avons expliqué dans 13.3.6.

Le principe de variables de vinaigre est que la fonction trappe ne peut être inversée par la personne qui possède la clef secrète que si la valeur des variables de vinaigre est fixée. Sinon, même avec la clef secrète il est difficile de trouver une solution.
Il va de soi que dans la clef publique ces variables de vinaigre ne sont pas un sous ensemble de tous les variables, mais qu'elles sont cachées par deux transformations affines aléatoires.

♣ **HFEv-** consiste à enlever un certain nombre d'équations publiques de HFEv.

♣ **HFEf** L'opération 'f' consiste à fixer quelques variables dans les équations publiques.

♣ **HFE+** L'opération '+' consiste à ajouter quelques équations aléatoires aux équations publiques de HFE.

♣ **HFE-+, HFEf+, HFEv-+ etc..** On construit de façon analogue de nombreuses autres variantes. Après chacune des transformations, il faut "ré-mélanger" les équations avec deux transformations affines aléatoires S' et T'.

17.1.1 Utilisation des variantes de HFE

Les HFE– et HFEv ne peuvent servir au chiffrement, à cause de la compression (et donc perte) d'information qui est introduite. Par contre elles peuvent être utilisées en signature, où le fait d'avoir plusieurs solutions n'est pas gênant. Les variantes HFE+ et HFEf expandent l'information, peuvent servir en chiffrement et ne peuvent pas servir en signature. Tout cela n'est bien entendu qu'une simplification, dans les cas particuliers il peut en être autrement. De plus combiner les opérations peut restaurer l'équilibre entre le nombre de variables à l'entrée et à la sortie.

On remarque que les 4 opérations évoquées v,f,+,– ne commutent pas en général entre elles. Elle ne commutent pas non plus avec d'autres fonctions, composantes de la génération d'une paire (clef publique, clef secrète) d'un schéma multivariable, par exemple avec l'opération qui consiste à mélanger des variables par un changement de variable linéaire S. On peut donc proposer une très grande quantité de schémas possibles qui combinent ces opérations.

17.2 La rapidité de HFE

17.2.1 Rapidité en clef secrète.

On peut dire de manière générale que

- Dans la version algébrique, de base, de HFE, les opérations de déchiffrement ou signature sont assez lentes.
  Par exemple, pour le HFE Challenge 1 de 13.1, décrit en détails dans 12.2.1 on a $K = \mathbb{F}_2, n = 80$ et $d = 96$. Il faut alors 1.20 s pour calculer 1 seul inverse de la fonction trappe HFE sur un Pentium III à 500 MHz. Dans le cas de Quartz on a $\mathbb{F}_2, n = 100, d = 129$, ce qui donne le temps de calcul de 3.6 secondes par inverse [66]. Des nombreux autres exemples peuvent être trouvés à `http://hfe.minrank.org`
- Les versions 'combinatoires', modifiées de HFE telles que HFEv décrites dans 17.1 ou [7] nécessitent des paramètres beaucoup plus petits que la version algébrique, et peuvent être de milliers de fois plus rapides, mais leur analyse de sécurité repose sur des bases moins solides.

17.2.2 Rapidité en clef publique

Tous les HFE sont extrêmement rapides en clef publique (ou en vérification de signature). Les calculs de polynômes quadratiques peuvent être effectués très rapidement en temps proportionnel à la longueur de la clef publique (qui est de l'ordre de n^3). Dans la cas binaire on peut en plus faire 64 additions binaires en parallèle avec un XOR de 64 bits.

17.3 HFE en chiffrement

De manière générale le problème de rapidité évoqué ci-dessus pour HFE n'a que très peu d'importance en chiffrement. En effet, dans l'utilisation standard des fonctions trappe en chiffrement, on chiffre seulement une clef de session choisie au hasard, et ensuite on utilise cette clef de session avec des moyens de chiffrement classiques (chiffrement par blocs tel que DES ou IDEA). Il n'est a priori pas gênant que le chiffrement de la clé de session, effectué une fois par session, soit assez lent. Pour les mêmes raisons le taux de transmission d'un cryptosystème à clef publique a assez peu d'importance en utilisation pratique pour des longs messages.

17.3.1 Les valeurs conseillées

On va peut conseiller de prendre des valeurs bien au delà des attaques connues, par example $n \geq 160$, $d > 96$. De plus on utilisera une variante de HFE, par exemple HFEf décrite dans 17.1 ou [7]. Dans ce ca là on peut se permettre $d = 25$.

17.4 HFE en signature

Les valeurs conseillées

On peut conseiller aussi $n > 160$, $d > 96$ et on utilisera une variante qui compresse l'information, par exemple HFEv- [76, 7, 65]. Si l'application exige que les signatures doivent être rapides, il faut alors prendre un HFEv-, jamais un HFE pur, avec $n \geq 128$ et $d \geq 25$.

On peut ainsi obtenir des systèmes de milliers de fois plus rapides que RSA, par exemple le C^{*--} dont la sécurité est analysée dans [59] et qui bat tous les records de vitesse. Cela est utilisé dans le schéma Flash [61]. Cependant il faut noter que la sécurité de ce type de systèmes est assez controversée.

17.4.1 Les signatures record

HFE est un des rares cryptosystèmes connus qui permettent de faire des signatures très courtes, de l'ordre de $80 - 128$ bits. Cela est lié bien entendu au fait que les meilleures attaques sur HFE restent toujours proches de la recherche exhaustive mais cela n'est pas suffisant. Pour obtenir des signatures courtes il faut aussi un protocole de signature modifié par rapport à celui habituellement employé. La partie qui suit est consacrée entièrement à de telles protocoles appelés "schémas de signature".

Par contre il semble difficile (et risqué) d'obtenir un schéma qui donne des signatures qui soient à la fois rapides et courtes.

17.5 Brevets

L'inventeur de HFE est Jacques Patarin. Les brevets sont propriété de Schlumberger CP8, anciennement Bull CP8. Le brevet nord-américain est connu en tant que US Patent 5,790,675 et expire le 24/07/2016. On peut le consulter à l'adresse http ://www.uspto.gov/patft/. HFE est aussi breveté en France sous numéro 2737370 (expiration 27/07/2015) ainsi qu'en Israël, Corée du Sud et à Taiwan. Des demandes de brevets ont également été déposées dans 8 autres pays.

Chapitre 18

Signatures courtes

On peut toujours raccourcir une signature en rallongeant sa vérification. Avant d'explorer ce trade-off dans 18.6 on va d'abord étudier la longueur des signatures dont le vérification est très rapide : composée des calculs d'une fonction trappe F et des applications d'une fonction de hachage H.

18.1 La solution classique

On commence par étudier la façon habituelle de faire des signatures. Soit F une fonction trappe sur $\mathbb{F}_q^n \to \mathbb{F}_q^m$. Le schéma de signature classique utilise une fonction de hachage sans collision H et calcule la signature σ d'un message m comme :

$$\sigma = F^{-1}(\ H(m)\ )$$

18.1.1 Falsification Existentielle (Existential Forgery)

Malheureusement il y a l'attaque suivante utilisant le paradoxe d'anniversaires

1. Générer $q^{m/2}$ **versions** du message à signer $m_1, \ldots, m_{2^{n/2}}$, en jouant sur les virgules, les espaces, pièces jointes etc..
2. Générer une liste de $2^{m/2}$ valeurs $F(\sigma_j)$, pour les σ_j connus.
3. Avec une probabilité proche de 1/2 il existe (i, j) tels que $F(\sigma_j) = H(m_i)$.
4. Pour les trouver en $2^{m/2}$ opérations, insérer les éléments de deux listes dans une table de hachage.

Un schéma de signature avec longueur de 80 semble donc difficile à réaliser : le schéma standard est alors cassé avec environ 2^{40} calculs et autant de mémoire. Plus généralement, dans le Théorème 18.3.0.2 on montre que l'on peut casser n'importe quel schéma de signature basé sur un seul calcul de F^{-1} en $q^{n/2}$.

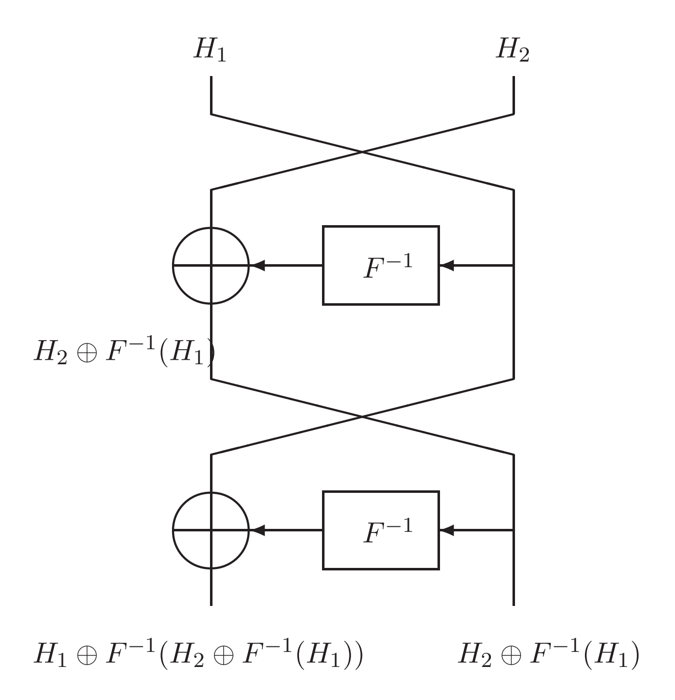


FIG. 18.1 – Génération des signatures avec le schéma de Feistel

18.2 Les signatures courtes $m = n$

Dans tous nos schémas de signature sui suivent on notera par $F^{-1}(y)$ un quelconque inverse choisi au hasard. Dans un premier temps on suppose que $m = n$ et que, ou bien la fonction (F est bijective, ou bien simplement on ne sera capable de signer qu'une partie des messages. Ainsi on ignore les cas où $F^{-1}(y)$ n'existe pas qui sera étudié dans 18.4.1.

18.2.1 Le Schéma de Feistel-Patarin avec $m = n$

Le premier exemple d'un protocole qui utilise plusieurs F^{-1} pour éviter l'attaque en $q^{n/2}$ et obtenir des signatures courtes a été proposé par Jacques Patarin dans [65].

Soit $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ une fonction trappe avec $m = n$. Soit H une fonction de hachage et $H_i(m) = H(i||m)$. On calcule la signature σ comme suit :

$$\boxed{\sigma = F^{-1}\left(\;\; H_1 + F^{-1}\left[H_2 + F^{-1}(H_1)\right] \;\; \right)}$$

Cette méthode directement inspirée du schéma de Feistel avait été proposée et étudiée par Jacques Patarin dans la version étendue de [65]. En effet, cette équation peut être obtenue avec un schéma de Feistel comme le montre la Figure 18.1.

La meilleure attaque connue contre ce schéma de signature est en $q^{3n/4}$ [65, 7]. Dans le théorème 18.3.0.2) on va obtenir une formule beaucoup plus générale.

Dans un cadre général, on va appeler ce schéma le schéma de Feistel-Patarin, qui sera défini comme suit pour un nombre quelconque d'itérations K.

```
σ ← 0
for i = 1 to K do
    {
        σ ← σ ⊕ H_i(m)
        σ ← F^{-1}(σ)
    }
return σ
```

TAB. 18.1 – Le schéma de Feistel-Patarin avec $m = n$ et K inverses

On remarque qu'ici on utilise des valeurs H_i différentes à chaque itération. C'est plus prudent. Suivant une attaque décrite par Patarin dans la version étendue de [65], la sécurité de ce schéma est de $q^{\frac{K}{K+1}m}$. En théorie on peut donc s'approcher de q^n aussi près qu'on le souhaite en ajoutant encore des itérations (avec des calculs F^{-1} supplémentaires). Cela donnerait des schémas de signature qui ne sont pas cassés avec des longueurs de 80 ou 128 bits. En pratique, ce n'est pas aussi simple : souvent la fonction F n'est pas bijective et en pratique on pourra pas signer tous les messages. Randomiser la fonction inverse pose des problèmes de vérification : il faut garder tous les bits d'aléa ajoutés afin de pouvoir vérifier la signature. Cela rallonge des signatures. Pour toutes ces raisons on va être amené à étudier des schémas de signature beaucoup plus généraux.

18.3 Des schémas de signature plus généraux

On va décrire une classe beaucoup plus large de schémas de signature qui généralisent le schéma de Feistel-Patarin avec n'importe quelle fonction trappe $F : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$. Puisque dans la cas général il ne sera peut-être pas possible d'inverser la fonction en tout point, on suppose que à chaque étape on a $R \geq 0$ bits de randomisation de sorte à avoir 2^R essais possibles sur F^{-1}. Il y a essentiellement deux interprétations possibles pour R.

1. Si $m > n$, le schéma de signature peut permettre plusieurs choix possibles pour certains bits de y, afin d'obtenir une solution x. Dans ce cas y est récupéré en entier à partir du x au moment de la vérification et la longueur de la signature ne sera pas rallongée à cause de R.
2. Si $m \leq n$ et si $m << n$ R n'est pas nécessaire, mais si $n - m$ est petit il va falloir hacher quelques bits supplémentaires dans le protocole de signature, afin d'avoir possibles 2^R essais à faire qui permettront de signer plus de messages. A ce moment-là les R bits de la randomisation doivent être présents dans la signature. Sinon, il est impossible de la vérifier. Le problème d'omettre de bits dans une signature est étudié séparément dans 18.6.

En principe on ne suppose pas (encore) que le schéma permet de signer tous les messages. Pour le calcul de la valeur minimale de R qui le permet, voir 18.4.1. Par contre La valeur de R va jouer un rôle important dans les attaques comme le montrera le Théorème 18.3.0.2. Elle va aussi ralentir le calcul d'une signature, car (dans le pire cas) il faudra faire environ 2^R essais.

Definition 18.3.0.1 (Un (n, m, K, R)-schéma de signature) *Soit $F : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ une fonction trappe. On appelle un (n, m, K, R)-schéma de signature n'importe quel schéma vérifiable sous forme générale suivante :*

1. *Pour calculer une signature doit faire K fois les opérations suivantes :*
   (a) *Fixer R bits de randomisation présents à chaque étape qui permettent de répéter l'ensemble de l'étape 2^R fois au cas d'échec.*
   (b) *L'étape consiste à* **un** *calcul de un F^{-1} et une combinaison avec des hachés et des résultats des étapes précédentes.*
2. *La signature est calculée comme une combinaison quelconque des données de toutes les étapes et inclut les RK bits de randomisation nécessaires pour la vérification. On peut omettre une partie de ses données qui est recalculée au moment de vérification, ce qui augmente le temps de vérification.*

Exemple : Quartz publié dans [66, 67] est un $(107, 100, 4, 0)$-schéma de signature construit selon la méthode particulière décrite dans 18.4.3.

Théorème 18.3.0.2 *Soit $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ une fonction trappe. Soit Σ un (n, m, K, R)-schéma de signature. Alors on peut forger une signature en 2^A d'opérations CPU, avec essentiellement 2^B de mémoire pour tout $A \geq B > 0$ tels que*

$$A + KB + (K-1)R = Km \log_q 2$$

Preuve : On calcule l'image de F pour 2^B valeurs aléatoires que l'on insère dans une table de hachage. Cela permet d'inverser F avec probabilité $2^{B-m\log_q 2}$ pour un y aléatoire. On va démarrer avec une population de 2^{A-R} messages qui en seront pas stockés mais essayés à tour de rôle. En comptant la première randomisation on aura 2^A essais à faire. Après la première étape on arrivera à inverser $2^{A+B-m\log_q 2}$ parmi tous ces essais. Maintenant pour chacune des étapes $2..K$ on va d'abord augmenter la population 2^R fois au moment de la randomisation, et ensuite la réduire $q^{B-m\log_q 2}$ fois en inversant F. La population finale devrait être de 1 en moyenne ce qui donne :

$$2^{A+B-m\log_q 2+(K-1)(R+B-m\log_q 2)} = 1$$

$$A + KB + (K-1)R = Km \log_q 2 \quad \square$$

Corollaire 18.3.0.3 *Si l'on considère que le prix du temps est le prix de la mémoire alors on a une attaque en temps et mémoire égaux à :*

$$q^{\frac{Km}{K+1} - \frac{R(K-1)\log_2 q}{K+1}}$$

Par example pour Quartz décrit dans [66, 67] on a $m = 100$, $n = 107$, $R = 0$, $K = 4$ et cela donne la sécurité en 2^{80}.

Problème ouvert : Comment conduire cette attaque avec moins de mémoire ?

Dans la pratique le frein principal n'est pas le temps de calcul, mais la mémoire. Si l'on suppose que le temps de calcul est environ le carré de la mémoire disponible pour le même prix, on doit avoir $A \geq 2B$. Cela va nous amener à avoir $A = 2B$ sauf si un tel A s'avère trop grand par rapport aux autres attaques. Assez souvent la recherche exhaustive sur F qui est en $q^{min(m,n)}$ et sans mémoire, sera plus rapide avec $2^A > q^{Min(m,n)}$. On a donc :

Corollary 18.3.0.4 *On suppose que le coût du temps de calcul est environ le carré de la mémoire disponible pour le même prix.*

Soit

$$A = \frac{2Km\log_2 q}{K+2} - \frac{2R(K-1)}{K+2}$$

Si $2^A > q^{Min(m,n)}$ *alors l'attaque la moins chère sera la recherche exhaustive en* $q^{Min(m,n)}$ *et sans mémoire, sinon on a une attaque en temps*

$$q^{\frac{2Km}{K+2} - \frac{2R(K-1)\log_2 q}{K+2}}$$

et en mémoire de

$$q^{\frac{Km}{K+2} - \frac{R(K-1)\log_2 q}{K+2}}$$

Cette évaluation plus réaliste que la précédente, donne un tout autre résultat pour Quartz décrit dans [66, 67] on a $m = 100$, $n = 107$, $R = 0$, $K = 4$. Le calcul donne $A = 133$, ce qui moins rapide que la recherche exhaustive sur F en 2^{100}. Cela montre que Quartz ne sera probablement pas cassé avant 2020.

18.4 Extensions du schéma de Feistel-Patarin

Le schéma initial de Feistel-Patarin décrit ci-dessus est prévu uniquement pour $m = n$. Il est intéressant de disposer d'un schéma de signature général. Par example on a assez souvent $n > m$ dans les variantes les plus difficiles de HFE comme $HFEv-$, voir 17.1 et [76, 7, 65].

18.4.1 Comment signer tout message

La notion de (n, m, K, R)-schémas de signature introduite dans 18.3 prévoit un paramètre de randomisation R qui permet de s'assurer que dans tous les cas on arrive à fabriquer une signature, sauf avec une probabilité négligeable. Dans certains cas R n'est clairement pas nécessaire, par example si $n >> m$, et plus précisément dans les cas décrits dans 18.4.3.

On va supposer que la fonction trappe utilisée se comporte comme une fonction aléatoire $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$. Cette hypothèse est souvent vraie en pratique. Elle a été vérifiée expérimentalement pour HFE dans [58], et souvent peut être prouvée dans différents contextes, voir par example [57], page 102, exercice 5.

Proposition 18.4.1.1 *Avec une fonction aléatoire* $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ *on peut atteindre une probabilité d'échec inférieure à* 2^{-80} *pour inverser* F *pour :*

$$R = \begin{cases} 6 & \textit{if} \quad m \geq n \\ \lceil 5.79 - (n-m)\log_2 q \rceil & \textit{if} \quad n > m \geq n - \frac{5.79}{\log_2 q} \\ 0 & \textit{if} \quad m < n - \frac{5.79}{\log_2 q} \end{cases} \tag{18.1}$$

Preuve : Soit $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ une fonction aléatoire avec $m = n$. Il est facile à montrer que si $n \to \infty$ la probabilité que le cardinal de $F^{-1}(\{y\})$ soit égal à i pour un y aléatoire suit la loi $Pr(i) = \frac{1}{ei!}$. La probabilité que y ne soit pas dans l'image de F est donc de $1/e$. Environ 63.2% des y sont dans l'image de F pour $m = n$.

On va regarder tous les 3 cas :

1. Cas $m \geq n$.

   On va se permettre d'introduire (n'importe où) dans le calcul de y à inverser R bits aléatoires. Ces R bits doivent être ajoutés dans la signature pour la vérification. Puisque $m \geq n$ dans le pire des cas la probabilité qu'il n'y ait pas d'inverse pour un y donné sera $\frac{1}{e}$. La probabilité qu'il n'y ait pas de solution dans pour aucune de 2^R possibilités est e^{-2^R}. Pour avoir $e^{-2^R} < 2^{-80}$ il faut :

$$R > \log_2(80 \ln 2) = 5.79$$

2. Cas $n > m \geq n - \frac{5.79}{\log_2 q}$.

   On va regarder F comme $q^{n-m} < 2^{5.79}$ functions aléatoires $\mathbb{F}_q^m \to \mathbb{F}_q^m$. La probabilité pour une telle function et un seul y, $F^{-1}(y)$ n'ait pas de solution est $\frac{1}{e}$. On 2^R possibilités pour y et q^{n-m} functions. Donc la probabilité qu'il n'y ait jamais de solution est $e^{-2^R - q^{n-m}}$. Pour avoir $e^{-2^R q^{n-m}} < 2^{-80}$ il faut :

$$2^R q^{n-m} > 80 \ln 2$$

$$R > 5.79 - (n-m)\log_2 q$$

3. Cas $m < n - \frac{5.79}{\log_2 q}$.

   Dans ce cas on va encore regarder F comme q^{n-m} functions aléatoires $\mathbb{F}_q^m \to \mathbb{F}_q^m$, mais cette fois-ci $q^{n-m} > 2^{5.79}$. Cela implique que la randomisation n'est pas nécessaire car $e^{-q^{n-m}} < 2^{-80}$. □

Remarques

Dans le schéma Quartz décrit dans [66, 67] et qui fonctionne selon 18.4.2, on a $R = 7$. Cela est du au fait que dans Quartz on exige que F ait exactement 1 inverse, afin de rendre la signature déterministe. A ce moment là il est facile à voir qu'il faut remplacer 5.979 et 6 par respectivement 6.92 et 7.
On pourrait également mettre 5 à la place de 5.979 et de 6, à condition d'accepter une probabilité plus grande d'échec de 2^{-46} ce qui est souvent acceptable.

Important :

Dans certain cas on est amené à prendre un R plus grand que celui que suggère la condition 18.1. Par example si la fonction est de type HFE-+, avec $m = n$, $r = 5$ équations supprimées qui ont été remplacées par 5 équations aléatoires (voir 17.1), la personne qui possède la clef secrète n'est capable d'inverser la fonction qu'avec une probabilité de l'ordre de 2^{-5}. Il faudra alors allonger R ce qui d'après 18.3.0.2 va dégrader la sécurité.

18.4.2 Feistel-Patarin généralisé avec $m \geq n - \frac{5.79}{\log_2 q}$

Soit $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ une fonction trappe avec $m \geq n - \frac{5.79}{\log_2 q}$.
Soit H une fonction de hachage et $H_i(m) = H(i||m)$. soir R défini par 18.1. On calcule une signature comme suit :

```
σ ← 0
for i = 1 to K do
    for all(Rand_i ∈ {0,1}^R) do
    {
            σ ← σ ⊕ H_i(m)
            σ ← F^{-1}(σ||Rand_{i1}||...||Rand_{iR})
    }
return σ||Rand_{11}||...||Rand_{KR}
```

Tab. 18.2 – Le Feistel-Patarin avec $m \geq n - \frac{5.79}{\log_2 q}$

D'après 18.1 chaque itération va pouvoir signer avec une probabilité de $1 - 2^{-80}$. La probabilité d'échec globale sera de l'ordre de $1 - K2^{-80}$. Ce type de schéma est assez lent. Soit $T_{F^{-1}}$) le temps nécessaire pour calculer F^{-1} et soit $DT_{F^{-1}}$ le temps nécessaire pour décider qu'il n'y en a pas pour un y donné. Le temps d'une signature sera de l'ordre de

$$K2^R \cdot DT_{F^{-1}} + K \cdot T_{F^{-1}}.$$

La longueur de la signature sera de

$$|\sigma| = \lceil n \log_2 q \rceil + KR \quad \text{bits}$$

On va donner un example de signature la plus courte que l'on puisse obtenir avec $R = 6$, $q = 2$ et dans le cadre réaliste avec une sécurité de 2^{80} calculs et $< 2^{40}$ en mémoire. On vérifie alors que le Corollaire 18.3.0.4 donne ce niveau de sécurité pour $K = 2$, $m = 86$ et $n = 80$. Cela donne des signatures de 92 bits.

18.4.3 Feistel-Patarin généralisé avec $m < n - \frac{5.79}{\log_2 q}$

Soit $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ une fonction trappe avec $m < n - \frac{5.79}{\log_2 q}$, ce qui d'après 18.1 permet de pouvoir signer avec une probabilité de $1 - 2^{-80}$ avec $R = 0$. Soit $\lambda \in \mathbb{F}_q^l$. On note par $[\lambda]_{r \to s}$ le sous-string :

$$[\lambda]_{r \to s} = (\lambda_r, \lambda_{r+1}, \ldots, \lambda_{s-1}, \lambda_s).$$

```
σ ← 0
for i = 1 to K do
{
        σ ← σ ⊕ H_i(m)
        U ∈ F^{-1}(σ)
        σ ← U_{1→m}
        Add_{i1}||...||Add_{i(n−m)} ← U_{(m+1)→n}
}
return σ||Add_{11}||...||Add_{K(n−m)}
```

TAB. 18.3 – Le Feistel-Patarin avec $m < n - \frac{5.79}{\log_2 q}$

Le temps de la signature est essentiellement le temps de calculer K inverses de F. Le fait de tronquer rallonge la signature de $K \cdot (n - m)$ ce qui donne

$$|\sigma| = \lceil ((m + K \cdot (n - m)) \log_2 q \rceil \quad \text{bits}$$

Ce schéma est appliqué dans :

Le schéma Quartz

Dans le schéma Quartz soumis au projet Européen Nessie [66, 67], nous avons $q = 2$, $n = 107$, $m = 100$ et $K = 4$, ce qui donne des signatures de 128 bits avec le niveau de sécurité de 2^{80}.
Quartz contient des détails supplémentaires qui le rendent déterministe [66, 67].

18.5 D'autres solutions

18.5.1 Le degré 3

Jacques Patarin avait également proposé une toute autre façon de faire des signatures courtes avec des polynômes multivariables. Cette méthode est décrite dans [57, 60] et utilise un polynôme sous forme $F(x_1, \ldots, x_n, h_1, \ldots, h_n)$, de degré total 2 en x_i et de degré total 1 en h_i. Une signature valable doit satisfaire essentiellement $F(\sigma, H(m)) = 0$.

18.5.2 La signature différentielle

Cette (autre) solution au problème de la signature courte est assez surprenante et avait été proposée par l'auteur. Pour un nombre d'inverses constant K dont dépend la rapidité, elle permet d'avoir de signatures plus courtes et d'avoir la valeur de $(m-n)$ deux fois plus importante que ci-dessus.

Soit $F : \mathbb{F}_2^{m+\delta} \to \mathbb{F}_2^m$ une fonction trappe HFE avec δ un petit entier avec $\delta^2/2 < m$.

Nous allons expliquer ce schéma de signature directement sur un exemple de telle fonction, $F : \mathbb{F}_2^{128} \to \mathbb{F}_2^{120}$ qui pourrait être construite comme un HFEv- avec $v = 1$ et $r = 7$.[1]

On calcule la signature σ comme :

$$\boxed{\sigma = F^{-1}(H_2) - F^{-1}(H_1)}$$

La vérification de la signature est un peu plus complexe :

1. On cherche un X tel que :
$$\begin{cases} F(X) &= H_1 \\ F(X+S) &= H_2 \end{cases}$$
2. La différence entre ces deux équations donne 120 équations **linéaires** avec 128 variables X_i.
3. Celles-ci donnent une solution paramétrique qui permet de réécrire l'équation $F(X) = H_1$ comme 120 équations quadratiques avec 8 nouvelles variables a_i.
4. Un tel système est linéaire après l'introduction de $8 * 9/2 = 36$ nouvelles variables $\alpha_{ij} = a_i a_j$ comme dans 6.2.4.
5. On résout le système de 120 équations avec $36 + 8$ variables α_{ij} et a_i par la réduction de Gauss.
6. On refait le changement de variables pour obtenir les valeurs des variables d'origine X_1.
7. Le fait d'avoir pu trouver une solution au $\begin{cases} F(X) &= H_1 \\ F(X+S) &= H_2 \end{cases}$ signifie que la signature est correcte.

Ce schéma donne des signatures de 128 bits avec un facteur de travail de 2^{80} exactement. La même méthode marche pour $n = 120$ jusqu'à $n = 133$.

18.5.3 La signature différentielle améliorée.

Il s'agit d'une généralisation du schéma précédent, qui donne les même tailles de signatures pour le même facteur de travail, car elle utilise également deux inverses. Un avantage de cette variante est qu'elle utilise une taille variable de la valeur de hachage H_2. On l'expliquera sur le même exemple, $F : \mathbb{F}_2^{128} \to \mathbb{F}_2^{120}$.

[1]Dans ce cas le basic HFE interne fonctionne sur $\mathbb{F}_2^{127}$, ce qui donne une extension première

Soit $H_2(m)$ une valeur de hachage du message m sur k bits. Soit $(x,h) \mapsto B(x,h)$ une fonction bilinéaire $F : \mathbb{F}_2^{128} \times \mathbb{F}_2^k \to \mathbb{F}_2^{120}$. Elle est aléatoire, mais on n'a pas besoin de la stocker car elle peut être définie à partir d'une graine de 128 bits avec une fonction d'expansion pseudo-aléatoire en utilisant par exemple le DES.

Pour calculer une signature on calcule d'abord une solution X à :

$$X = F^{-1}(H_1)$$

Ensuite la signature σ est obtenue comme :

$$\boxed{\sigma = F^{-1}(B(X, H_2)) - F^{-1}(H_1)}$$

La vérification de la signature se fait comme suit :

1. On cherche un X tel que :

$$\begin{cases} F(X) & = & H_1 \\ F(X+S) & = & B(X, H_2) \end{cases}$$

2. Comme dans 18.5.2 on a 120 équations **linéaires** avec 128 variables X_i. Les étapes 2-7 de la vérification sont faites exactement comme dans 18.5.2.

Remarques :

On peut également calculer des signatures comme (3 inverses) :

$$\boxed{X = F^{-1}(H_1); \quad \sigma = F^{-1}(B(X, H_2)) - F^{-1}(B(X, H_2))}$$

(prendre toujours des racines différentes).

Autre version (4 inverses)

$$\boxed{X = F^{-1}(H_2) - F^{-1}(H_1); \quad \sigma = F^{-1}(B(X, H_3)) - F^{-1}(B(X, H_3))}$$

ou (5 inverses)

$$\boxed{X = F^{-1}(H_2) - F^{-1}(H_1); \quad X' = F^{-1}(B(X, H_2)) - F^{-1}(H_1)}$$
$$\boxed{\sigma = F^{-1}(B(X', H_3)) - F^{-1}(B(X', H_3))}.$$

18.6 Le compromis longueur / vérification

Pour tout schéma de signature il y a un compromis longueur des signatures - temps de vérification. Si on enlève t bits à la signature, on doit être prêt à faire 2^t essais pour la vérification. Pour de nombreux cryptosystèmes le problème de récupérer juste quelque bits d'une entrée x, étant donné la plupart des bits de x, est nettement plus facile que la recherche exhaustive. Les détails dépendent du cryptosystème. Par example dans [116] Matthieu Finiasz, Nicolas Sendrier et moi-même montrons comment faire des signatures courtes avec McEliece. Il s'avère que admettre une vérification en 2^{30} permet de raccourcir la signature de 60 bits environ [116].

18.7 La taille de l'espace des massages

Il y un autre moyen de raccourcir une signature qui a déjà été suggéré par Merkle [20]. Cela est possible si le cardinal des des messages possibles est petit, Par example si le message à signer est 0 ou 1, La longueur de signature peut être de 50 bits avec une sécurité de 2^{80}. Pour cela il suffit de servir d'une fonctions à sens unique f dont le calcul prend 2^{30} d'opérations. Le clef publique consistera en 2 valeurs obtenus au moment de la génération des clefs $f(x_0)$ et $f(x_1)$. La signature de 0 sera x_0 et la signature de 1 sera x_1. Des nombreuses généralisations de cette idée simple sont possibles. Un certain nombre de variantes sont décrites par Merkle dans [20].

Cinquième partie

Le problème IP

Chapitre 19

Autour du problème IP

Le problème désigné par IP dans la présente thèse est un cas particulier du problème général IP avec 2 secrets de [65], avec le nombre d'équations égal au nombre d'inconnues $m = n$ et les fonctions S et T linéaires.

19.1 Les problèmes IP et MP

Polynômes multivariables donnés :

Soit $K = \mathbf{F}_q$, un corps fini. On identifie $x \in \mathbb{F}_{q^n}$ avec $(x_1, \ldots, x_n) = \mathbb{F}_q^n$.
Pour simplifier les écritures on posera $a_0 = 1$ et $x_0 = 1$.

Soit $\mathcal{A} : \begin{cases} b_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} a_i a_j \\ \text{avec } k = 1..n \end{cases}$ et $\mathcal{B} : \begin{cases} y_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} x_i x_j \\ \text{avec } k = 1..n \end{cases}$

deux ensembles de n équations quadratiques avec n variables sur K.

Morphisme de Polynômes (MP)

On appelle un **morphisme de polynômes**, un double changement de variables défini par un couple d'endomorphismes (S, T) :

$S : \begin{cases} K^n & \rightarrow & K^n \\ (x_1, .., x_n) & \mapsto & (a_1, .., a_n) \end{cases}$ $T : \begin{cases} K^n & \rightarrow & K^n \\ (b_1, .., b_n) & \mapsto & (y_1, .., y_n) \end{cases}$

qui a la propriété de transformer $\mathcal{A}$ en $\mathcal{B}$:

$$\mathcal{B} = T \circ \mathcal{A} \circ S$$

Isomorphisme de Polynômes (IP)

On dit que $\mathcal{A}$ et $\mathcal{B}$ sont **isomorphes** s'il existe un morphisme (S, T) avec S et T inversibles.

Le problème IP

Étant donnés les deux ensembles d'équations $\mathcal{A}$ et $\mathcal{B}$ isomorphes, trouver un isomorphisme S et T.

19.1.1 Remarques et problèmes annexes

Dans [65] on trouvera les énoncés d'autres versions du problème IP.

Nous avons posé $m = n$. C'est le cas 'central', en effet si $m = 1$ le problème est facile à résoudre (p.ex. voir chapitre 6 de [60]). De même pour $m > n^2/2$ la famille des équations engendre l'ensemble de toutes les équations quadratiques le problème devient trivial.

Le fait d'avoir séparé les cas linéaires et affines de IP, demande de supposer que $\mathcal{A}(0) = 0$ et $\mathcal{B}(0) = 0$. Sinon on introduit une faiblesse artificielle au problème : on a $t(\mathcal{A}(0)) = \mathcal{B}(0)$ et cela donne un savoir *a priori* sur T.

19.2 Applications de IP

IP est relié à la sécurité de quasiment tous les schémas multivariables. Le rôle joué par IP dans la sécurité de HFE est expliqué dans 13.2.
Dans [65] Jacques Patarin a proposé un schéma d'authentification à divulgation nulle de connaissance (Zéro-knowledge) ainsi qu'un schéma de signature correspondant, et cela à base du problème IP.
Malheureusement nous avons prouvé dans [88] que IP n'est pas NP-complet. De plus mes nouvelles attaques sur IP en $q^{n/2}$ [88, 58] ont montré que IP est beaucoup moins solide qu'initialement prévu, comparé aux meilleures attaques connues auparavant en $q^{n\sqrt{n}}$.

19.3 Introduction aux attaques de IP

Nous avons trouvé plus de quatre méthodes générales pour résoudre IP. Elles sont introduites dans [88] et décrites avec un grand nombre de détails dans la version étendue de [58].

Nous allons ici seulement expliquer leur principe.

19.3.1 Le va-et-vient (to-and-fro)

L'idée de l'attaque est de partir de quelques équations initiales qui relient les entrées de $\mathcal{A}$ et les entrées de $\mathcal{B}$ qui leur correspondent par l'isomorphisme (S, T).

Voilà un exemple concret sur $\mathbb{F}_2^5$ dans lequel deux valeurs 1 et 7 à l'entrée de $\mathcal{A}$ correspondent (par S) à deux entrées de $\mathcal{B}$ qui valent respectivement 1 et 2 .

$$\begin{cases} \quad \mathcal{B} & & \mathcal{A} \\ S(1) & = & 1 \\ S(2) & = & 7 \end{cases}$$

Comme S est linéaire, nous en déduisons une équation supplémentaire sur S. En tout, cela donne 3 équations linéairement dépendantes :

$$\begin{cases} \mathcal{B} & & \mathcal{A} \\ S(1) & = & 1 \\ S(2) & = & 7 \\ S(3) & = & 6 \end{cases}$$

L'étape suivante consiste à transformer les équations sur les entrées et sur S, en équations sur les sorties et sur T. Le diagramme suivant en explique le principe :

$$\begin{array}{cccccccccc|c} & T & \circ & \mathcal{A} & \circ & S & & = & \mathcal{B} & & \\ & & & & 1 & & 1 & & & & S(1) = 1 \\ & & & \swarrow & & & & & \searrow & & \Downarrow \\ 5 & & 1 & & & & & & & 5 & T(1) = 5 \end{array}$$

Chaque équation sur S permet d'obtenir une équation sur T.

Mais comme les $\mathcal{A}$ et $\mathcal{B}$ sont non-linéaires, les équations linéairement **dépendantes** sur S donnent des équations linéairement **indépendantes** sur T.

$$\begin{cases} \mathcal{B} & & \mathcal{A} \\ 5 & = & T(1) \\ 16 & = & T(4) \\ 24 & = & T(23) \end{cases}$$

Le miracle : en partant de 2 équations nous avons obtenu 3 équations.

Il va de soi que le va-et-vient peut continuer dans le sens opposé (moyennant l'inversion de $\mathcal{A}$ et $\mathcal{B}$) et produire de plus en plus d'équations et récupérer S et T à la fin.

Ainsi on sait casser IP en q^{2n}, ce qui est la complexité pour deviner les deux équations initiales. Comment faire mieux ?

19.3.2 Attaques en $q^{n/2}$

Nous décrivons ici les 3 idées principales qui permettent d'obtenir une attaque en $q^{n/2}$ au lieu de q^{2n} que nous venons d'obtenir pour une attaque de type va-et-vient.

Amélioration 1 - Éliminer l'inversion Afin de construire de meilleurs algorithmes nous avons cherché un moyen d'obtenir à partir d'une information sur les entrées de $\mathcal{A}$ et $\mathcal{B}$ de l'information non plus sur les sorties mais également sur les entrées de $\mathcal{A}$ et $\mathcal{B}$. Ainsi, on n'a pas besoin de va-et-vient, donc pas d'inversion de $\mathcal{A}$ et $\mathcal{B}$, et ce n'est qu'ainsi qu'on pourra descendre en dessous de q^n.

Amélioration 2 - Démarrer avec 1 valeur On aimerait pouvoir commencer le va-et-vient avec une seule valeur au lieu de deux. Nous avons cherché un moyen d'obtenir à partir d'une seule entrée de $\mathcal{A}$ (et d'une seule entrée de $\mathcal{B}$ correspondante), une autre entrée de chaque côté qui soit préservée par l'isomorphisme, c'est à dire que les entrées qui se correspondent par S continuent à se correspondre après la dérivation d'une nouvelle valeur.

Ces deux considérations nous ont amené à définir la notion suivante :

Définition 19.3.2.1 (Fonction de gain, *boosting function***)** *On appelle une* **fonction de gain** *une fonction $F = (F_{\mathcal{A}}, F_{\mathcal{B}})$, chacune étant $K^n \to K^n$, qui est préservée par l'isomorphisme de polynômes, c'est à dire :*

Si $S(x) = a \quad et \quad \left\{ \begin{array}{lcl} F_{\mathcal{B}}(x) & = & x' \\ F_{\mathcal{A}}(a) & = & a' \end{array} \right.$

Alors $S(x') = a'$.

Nous avons construit 4 telles fonctions non-triviales et basées sur des principes différents. Elles utilisent le fait que les équations sont quadratiques et donc leur différentielle discrète est linéaire.

Amélioration 3 - Meet-in-the-middle Étant donné une fonction de gain non-déterministe sous une certaine forme particulière, il est possible de concevoir une attaque de type 'anniversaire' sur IP que nous avons décrit pour la première fois dans [88]. Ainsi on arrive à la complexité de $q^{n/2}$.

Des détails de l'attaque sont décrits dans [88, 58] et aussi, dans le chapitre suivant de la version longue (uniquement) du présent manuscrit. .

19.4 Résultats des attaques de IP

Nos attaques sur IP donnent les complexités de l'ordre de $q^{\mathcal{O}(n)}$. Les qualifier d'exponentielles est toutefois une affaire de point de vue, car la taille du secret dans IP est $\mathcal{O}(n^2)$, et donc la recherche exhaustive est en q^{n^2}.

C'est **beaucoup** plus grand que $q^{\mathcal{O}(n)}$. Par exemple pour une valeur réelle de $n = 80$ on compare 2^{40} à 2^{3200}.

Si N est le logarithme de la complexité de la recherche exhaustive, alors nos attaques seraient en $q^{\mathcal{O}(\sqrt{N})}$, ce qui est **sous-exponentiel**.

Notre meilleure attaque permet de résoudre IP en

$$n^{\mathcal{O}(1)} q^{n/2}$$

avec également $n^{\mathcal{O}(1)} q^{n/2}$ de mémoire.

La meilleure méthode connue avant [65], version étendue [8], était en $\mathcal{O}(q^{\sqrt{2n}\sqrt{n}})$. Par example pour $q = 2$ et $n = 80$, on est passé de 2^{360} à 2^{40}.

Chapitre 20

Le problème MP

20.1 Introduction

Le problème MP (Morphisme des Polynômes) défini déjà dans 19 est une généralisation du problème IP avec les applications S et T qui ne sont plus bijectives.

Plus généralement, on peut également considérer des polynômes sur un anneau non-commutatif R. On appellera ceci le **"problème MP non commutatif"**.

20.1.1 Exemple concret du problème MP.

Soient les deux ensembles d'équations suivants :

$$(A)\begin{cases} b_1 = a_1a'_1 \\ b_2 = a_2a'_2 \\ b_3 = a_3a'_3 \\ b_4 = a_4a'_4 \\ b_5 = a_5a'_5 \\ b_6 = a_6a'_6 \\ b_7 = a_7a'_7 \end{cases}$$

$$(B)\begin{cases} y_1 = x_1x'_1 + x_3x'_2 \\ y_2 = x_2x'_1 + x_4x'_2 \\ y_3 = x_1x'_3 + x_3x'_4 \\ y_4 = x_2x'_3 + x_4x'_4 \end{cases}$$

Le premier ensemble, c'est simplement 7 multiplications. Le deuxième correspond à la multiplication de 2 matrices 2x2 :

$$\begin{pmatrix} y_1 & y_3 \\ y_2 & y_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} \cdot \begin{pmatrix} x'_1 & x'_3 \\ x'_2 & x'_4 \end{pmatrix}$$

Le problème est de trouver deux transformations linéaires S et T :

$$(a_1, ..., a_7, a'_1, ..., a'_7) = S(x_1, ..., x_4, x'_1, ..., x'_4)$$

$$(y_1, ..., y_4) = T(b_1, ..., b_7)$$

Telles qu'on puisse obtenir $\mathcal{B}$ comme une composition de $T \circ \mathcal{A} \circ S$.

Notre couple $\mathcal{A}$ et $\mathcal{B}$ correspond au problème de multiplier deux matrices 2x2 avec seulement 7 multiplications. Ce problème avait été posé et résolu en 1969 par Strassen [97]. Voilà sa solution :

$$\begin{cases} a_1 = x_3 - x_4 \\ a_2 = x_1 + x_4 \\ a_3 = x_1 - x_2 \\ a_4 = x_1 + x_3 \\ a_5 = x_1 \\ a_6 = x_4 \\ a_7 = x_2 + x_4 \end{cases} \qquad \begin{cases} a'_1 = x'_2 + x'_4 \\ a'_2 = x'_1 + x'_4 \\ a'_3 = x'_1 + x'_3 \\ a'_4 = x'_4 \\ a'_5 = x'_3 - x'_4 \\ a'_6 = x'_2 - x'_1 \\ a'_7 = x'_1 \end{cases} \qquad \text{et} \qquad \begin{cases} y_1 = b_1 + b_2 - b_4 + b_6 \\ y_2 = b_4 + b_5 \\ y_3 = b_6 + b_7 \\ y_4 = b_2 - b_3 + b_5 - b_7 \end{cases}$$

A ce jour personne n'a établi quel était le nombre minimal de multiplications nécessaire pour multiplier deux matrices 3x3. La meilleure valeur connue est 23 [94].

Dans [88] nous montrons que le problème MP est NP-dur, et sous certaines hypothèses raisonnables IP n'est pas NP-dur. MP est donc probablement plus difficile que IP. Toujours reste-t-il que nos algorithmes pour résoudre IP sont exponentiels, et donc il n'est pas du tout exclu que des algorithmes analogues existent pour MP.

20.1.2 Applications :

Trouver S et T donne souvent une méthode nouvelle et intéressante pour calculer $\mathcal{B}$, dans l'exemple évoqué ci-dessus on arrive à l'algorithme de Strassen pour la multiplication matricielle.

Dans [92] un autre MP, résolu avec des ordinateurs très puissants a permis de trouver comment calculer le produit vectoriel avec seulement 5 multiplications.

Sixième partie

Le problème MinRank

Chapitre 21

Genèse du problème MinRank

21.1 Les problèmes difficiles des codes

21.1.1 Le problème SD

L'origine des problèmes étudiés dans cette partie est assez ancienne. En 1969, Dominic Welsh, au cours d'une conférence des mathématiques à Oxford [109], demande de trouver un algorithme efficace pour trouver le plus court cycle d'un matroïde[1][2] vectoriel[3], ce qui équivaut à trouver le mot du poids minimal dans un code linéaire. Malheureusement, ce problème a peu de chances d'être résolu en temps polynomial. Dans leur article de 1978, Berlekamp, McEliece and van Tilborg ont montré que le problème de décodage d'un code linéaire qui s'appelle SD (Syndrome Decoding), ou de façon équivalente le problème du poids minimal d'un espace affine, est un problème NP-complet [100] pour $q = 2$. Il l'ont également conjecturé pour le cas d'un espace linéaire, mais le prouver est resté longtemps un problème ouvert qui n'a été résolu qu'en 1997 par Vardy [108] qui étend également sa preuve pour tout corps fini $\mathbb{F}_q$. Notons que le problème SD de la présente thèse est parfois appelé MLD (Maximum Likelihood Decoding) par certains auteurs qui réservent le mot SD pour le problème de décision. Il est facile à voir que les deux problèmes sont est équivalents en pratique : on peut utiliser MLD pour résoudre SD et réciproquement.

Contrairement à certains problèmes NP-complets, garantis difficiles dans le pire cas, et parfois très faciles dans le cas moyen, SD semble très difficile aussi

[1]Mot dérivé de matrice.

[2]Un matroïde M est un ensemble de base E avec une famille de sous-ensembles C appelés cycles tels qu'aucun cycle ne contient un autre cycle, et si $C1 \neq C2$ sont des cycles, alors pour tout $e \in C1 \cap C2$ il existe un cycle $C3 \subset (C1 \cup C2) \setminus \{e\}$.

[3]Un matroïde vectoriel est le matroïde obtenu comme suit pour un ensemble de vecteurs quelconque donné $\{v_1, \ldots, v_r\}$. On a $E = [1..r]$ et les cycles sont exactement les ensembles d'indices $\{i_1, \ldots, i_s\} \subset E$ qui correspondent aux ensembles linéairement dépendants minimaux pour l'inclusion. Il est facile de voir que si le vecteurs sont considérés comme des colonnes de la matrice de parité d'un code correcteur, alors les cycles de ce matroïde sont exactement des supports des mots du code qui sont minimaux pour l'inclusion. Le support du mot du poids minimal est alors le cycle le plus court du matroïde induit.

dans le cas moyen (un code aléatoire). Dans un article récent d'Anne Canteaut et Florent Chabaud [102], un nouvel algorithme pour ce problème est proposé qui améliore tous les algorithmes existants, notamment celui de Stern [106] et Lee et Brickell [105]. Malgré plus de 30 ans d'effort de recherche dans ce domaine fondamental, le problème demeure très difficile - tous les algorithmes connus sont exponentiels.

21.1.2 Le problème SD et la cryptographie

La cryptographie à clef publique, se base souvent sur des problèmes d'arithmétique, souvent pas NP-complets, et pour lesquels on connaît des algorithmes sous-exponentiels, ou alors sur les problèmes ad-hoc qui peuvent être cassés à tout moment. Il serait évidemment bien de pouvoir baser des schémas et de preuves de sécurité sur des problèmes plus difficiles.

Pour le chiffrement, le problème est résolu depuis 1978, puisque le cryptosystème de McEliece, et ensuite celui de Niederreiter, donnent des schémas de chiffrement basés sur le problème du poids minimal. Par contre ce n'est qu'en 2001 que Matthieu Finiasz, Nicolas Sendrier et moi-même avons montré que l'on peut faire des signatures avec [116], aussi bien pour McEliece qu'avec la variante de Niederreiter [128]. Il semble que les codes de Goppa sont suffisamment nombreux (leur cardinal est évalué dans [29, 116]) et suffisamment optimaux pour ne pas pouvoir être distingués d'un code aléatoire.

Si on n'arrive pas à appréhender la structure de ces codes, et il ne reste qu'à attaquer le problème général de décodage SD, comme pour un code aléatoire, qui est NP-dur. Malheureusement, comme Gilles Brassard l'avait montré, il y a peu de chances qu'inverser une fonction trappe repose réellement sur un problème NP-complet [10, 7]. L'interprétation de ce résultat est critiquée par Koblitz dans [57]. Il semble aussi que cet obstacle théorique est inexistant quant aux schémas de signature.

Les problèmes de codage ont également de nombreuses applications dans le domaine d'authentification. C'est Sami Harari, qui a lancé l'idée en 1989 [135], avec un premier schéma, encore peu pratique et qui avait été cryptanalysé par Pascal Véron [138, 139]. Par la suite, Marc Girault dans [134] et Jacques Stern dans [136] ont poursuivi les tentatives. Enfin en 1993, Jacques Stern avait proposé une version à la fois pratique, performante et sûre [137]. Pascal Véron (G-SD) avait proposé dans [139] une version qui améliore le schéma de Stern d'environ 33%, mais Anne Canteaut avait montré dans [101] que cela était au prix des attaques en 2^{61} au lieu de 2^{70} pour le schéma de Stern.

De son côté Kefei Chen a proposé en 1996 un schéma d'authentification basé sur des codes de type rang inventés par Gabidulin [130, 131]. Ses paramètres ont du être révisées à la hausse suite au travail de Florent Chabaud et Jacques Stern [114]. Dans un article en préparation [110] on montre que 2 des algorithmes pour MinRank présentés dans la présente thèse sont, au moins dans certains cas, plus performants que l'attaque de Stern-Chabaud [114].

Tous ces schémas d'authentification à divulgation nulle de connaissance (Zéro-knowledge) permettent aussi de faire des signatures, voir 2.2.[6]. Des schémas de signature ainsi obtenus sont prouvablement sûrs mais malheureusement assez peu commodes en pratique. Comme nous l'avons expliqué dans 2.4 et 2.5, l'effort de recherche planétaire de 10 dernières années en cryptographie à clef publique est de **prouver** la sécurité (voir Prouver) relativement à des **scénarios d'attaque** (voir 2.4) aussi généraux que possible. On souhaite aussi se baser sur des problèmes vraiment difficiles. La combinaison du Zéro-knowledge et certains problèmes NP-complets donne des schémas parmi les plus solides que connaît la science. En effet, le Zéro-knowledge permet de prouver la sécurité contre n'importe quel stratégie d'attaque dans lequel un Vérifieur malhonnête (mais intelligent) interagit avec le Prouveur. C'est la classe d'attaques la plus générale que l'on arrive à concevoir. Et tout cela repose sur les problèmes fondamentaux, et sans doute fondamentalement difficiles, du poids (ou rang) minimal.

21.2 De SD à MinRank

Le problème du poids minimal admet une généralisation naturelle proposée par Gabidulin en 1985 [111]. Si un code est défini sur une extension de corps, alors on peut regarder un mot de code comme une matrice. On substitue alors le problème du poids minimal (le nombre de colonnes non nulles) par le problème de rank minimal (le nombre de colonnes linéairement indépendantes). On peut alors définir une distance de type rang, qui remplacera la distance de Hamming usuelle. On appelle Rank-SD le problème de décodage d'un code avec la distance rang.

Il se trouve que l'on peut réduire le problème Rank-SD au problème suivant : trouver une combinaison linéaire des matrices données qui soit de rang minimal. On appelle MinRank ce problème, suivant un article de 1996 [112] qui n'a pourtant rien à voir avec ni la cryptographie ni le codage.

On cherchant bien dans la littérature il s'avère que ce type de problème apparaît un peu partout, par example dans un article publié à Crypto'93, Don Coppersmith, Jacques Stern et Serge Vaudenay [77, 78] résolvent déjà un MinRank (sans le dire ni sans le savoir) pour cryptanalyser le schéma de signature birationnel de Shamir [75].

C'est aussi le même problème MinRank, qui est apparu dans l'attaque de HFE de Shamir et Kipnis, publiée à Crypto'99, sur la cryptanalyse de HFE, voir 13.2. Dans 24.5.2 et 13.2.6 on a pu considérablement améliorer cette attaque en proposant de résoudre autrement ce MinRank, d'ailleurs par le même méthode qu'utilisent déjà Coppersmith, Stern et Vaudenay. C'est aussi le même problème

qui, comme montré dans un article écrit avec Louis Goubin à Asiacrypt'2000, fait que le cryptosystème TTM n'est pas solide, car le rang est constant (et trop petit) [80, 79]. Enfin, le problème du rang minimal sur $\mathbb{Z}$ généralise le problème du poids minimal sur $\mathbb{Z}$, très voisin du célèbre problème de réduction de réseaux (ou treillis, *lattice* en anglais), problème très prisé en cryptanalyse. On peut aussi coder assez facilement d'autres problèmes sur $\mathbb{Z}$ comme MinRank, par example la factorisation. Il existe en fait en algorithme qui permet de coder comme MinRank n'importe quel ensemble d'équations multivariables de degré borné.

Comme souvent en science, le problème MinRank n'a rien de neuf. Ce qui est neuf c'est de s'apercevoir que MinRank montre une unité profonde et permet de mieux comparer de différents problèmes connus en codage, cryptographie et algorithmique.

Il est également hautement non-trivial de concevoir un nouveau schéma à clef publique qui ne soit pas cassé. Et encore moins trivial d'en proposer une implémentation pratique avec un bon niveau de sécurité. Mais avant de faire cela on va d'abord définir le problème MinRank et ses variantes.

Chapitre 22

Le problème MinRank

22.1 Définitions

Il semble que le nom de MinRank apparaît pour la première fois en 1996 dans un rapport très complet sur le problème en question [112]. On va définir et étudier MinRank dans un cadre très général, pour les matrices rectangulaires $\eta \times n$. On va supposer toujours $n \leq \eta$. Parfois on va se restreindre au cas $\eta = n$.

Les données

Soit R un anneau commutatif.
Soient $m+1$ matrices $\eta \times n$ à coefficients dans R :

$$M_0; M_1, \ldots, M_m$$

Le problème MinRank

Le problème MinRank(η, n, m, r, R) est de trouver (au moins une) solution $\alpha \in R^m$ telle que :

$$Rank(\sum_i \alpha_i M_i - M_0) \leq r.$$

En pratique on a souvent $M_0 = 0$, et le problème devient linéaire :

$$Rank(\sum_i \alpha_i M_i) \leq r.$$

22.1.1 D'autres problèmes

Dans [112] on définit aussi de façon analogue les problèmes MaxRank, Sing et NonSing. Par exemple le problème Sing est le MinRank$(n, n, m, n-1, R)$ avec le rang $r = n - 1$.

22.1.2 Les problèmes de décision.

On appelle DMinRank le problème décisionnel.
Il est facile de voir que si R est un anneau fini à q éléments, MinRank est calculatoirement équivalent à DMinRank. En effet, si on dispose d'un oracle qui résout le problème décisionnel, on va lui soumettre M_0 et un hyperplan de co-dimension 1 choisi au hasard dans l'ensemble des matrices engendrés par $M_1, \ldots, M_m$. Avec probabilité $1/q$ la réponse sera oui, et on sait alors que α appartient au sous espace en question, ce qui donne une équation linéaire sur les α_i (l'équation de l'hyperplan choisi). Après environ mq essais on pourra calculer α.

22.1.3 Version combinatoire du MinRank

Le problème MinRank tel que nous l'avons défini est un problème d'algèbre linéaire. Il peut être assez facilement décrit par les équations. C'est pour cela qu'on l'appelle parfois le MinRank algébrique. Par contre, il est possible de définir des problèmes de rang qui perdent ce caractère équationnel (ou algébrique), et deviennent des problèmes combinatoires.

Les données

Soit R un anneau commutatif.
Soient E,S,T de sous-ensembles de R.
Soit M_0 une matrice à coefficients dans E.
Soient m matrices $\eta \times n$ à coefficients dans S :

$$M_1, \ldots, M_m$$

Le problème MinRank

Le problème général MinRank$_($$E, S, T)(\eta, n, m, r, R)$ est de trouver (au moins une) solution $\alpha \in T^m$ telle que :

$$Rank(\sum_i \alpha_i M_i - M_0) \leq r.$$

Cette définition reprend en partie les notations du [112]. Le MinRank défini dans [112] est équivalent à la version décisionnelle de notre MinRank combinatoire avec $T = R$, et avec la restriction supplémentaire aux M_i qui auraient des supports disjoints pour $i = 0..m$.

Chapitre 23

La nature du problème MinRank

23.1 La difficulté théorique du MinRank

23.1.1 L'universalité du MinRank

Le MinRank est un problème universel, dans le sens qu'il peut encoder n'importe quel système d'équations diophantiennes. Plus précisément on a :

Théorème 23.1.1.1 (L'universalité du Déterminant, Valiant 1979) *Tout équation multivariable sur un anneau peut être codé comme le déterminant d'une matrice dont les entrées sont des constantes ou des variables.*

La preuve originale de Valiant est dans [113]. Une preuve plus simple est donnée dans [112]. Les deux méthodes donnent un algorithme polynomial qui code un ensemble d'équations quelconque comme le déterminant d'une matrice. Un tel déterminant correspond au MinRank avec rang $r = Min(\eta, n) - 1 = n - 1$.

Le corollaire immédiat, quand l'anneau en question est $\mathbb{Z}$ est :

Corollaire 23.1.1.2 *MinRank contient donc le Dixième Problème de Hilbert, qui est de résoudre une équation diophantienne sur $\mathbb{Z}$. De ce fait MinRank sur $\mathbb{Z}$ est indécidable.*

On va étendre ce résultat aux ensembles d'équations.

Théorème 23.1.1.3 (L'universalité du Déterminant étendue) *Tout ensemble d'équations multivariables sur un anneau peut être codé comme MinRank.*

Preuve : Pour un corps fini la preuve est déjà faite dans [112], on écrivant simplement le système comme une seule équation d'une de nombreuses façons possibles. On va faire une preuve dans le cas général.

Pour chacune des équations $i = 1..M$ on construit une matrice K_i, $k_i \times k_i$ dont les entrées sont des variables ou des constantes et telle que le déterminant de la matrice donne l'équation en question. Une méthode explicite pour ce faire est donnée dans [113] et [112]. Ainsi le rang de la matrice K_i est égale à $k_i - 1$ si et seulement si l'équation est vraie. Ensuite on pose :

$$L = \begin{pmatrix} K_1 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & K_M \end{pmatrix}$$

Il est facile de voir que le rang de L est égal à $\sum_{i=1}^{M} k_i - M$ si et seulement si toutes les équations sont satisfaites. □

Un corollaire immédiat :

Corollaire 23.1.1.4 *MinRank contient le problème MQ de résoudre un ensemble d'équations multivariables quadratiques étudié dans 6.*

Or MQ est NP-dur, comme nous l'avons montré dans 10.2, et donc :

23.1.2 MinRank est NP-complet

Corollaire 23.1.2.1 *MinRank est NP-complet pour $r = n^{\varepsilon}, \varepsilon > 0$ et η polynomial en n.*

Pour $r = n - 1$ la preuve est faite plus haut. Cela est vrai aussi quand $r = n^{\varepsilon}, \varepsilon > 0$, car le problème peut contenir une sous-matrice avec $(r+1) \times (r+1)$, pour laquelle c'est déjà NP-complet. En effet, la réduction triviale qui consiste à agrandir les matrices avec des zéros, est une réduction polynomiale car n et η sont polynomiaux en r.

Notre réduction (qui est aussi à peu près celle de [112]), peut toutefois paraître pas tout à fait satisfaisante, parce que dans les instances de MinRank ainsi construites, la taille des matrices n sera comparativement grande (polynomiale en) au nombre de matrices m. Fort heureusement il existe une autre preuve encore plus simple du fait que MinRank est NP-dur avec m et n de même ordre de grandeur. Elle est donnée dans 23.2.2.

23.1.3 MinRank et la factorisation

On va montrer comment construire une instance de MinRank qui soit aussi dure que la factorisation.

Méthode 1 On utilise le Théorème 23.1.1. Il suffit donc d'exprimer le problème de factorisation comme une équation. Ce sera l'équation $N = xy$, mais afin d'assurer que $x > 1$ and $y > 1$ on va représenter un entier non-négatif comme somme de quatre carrés. Un théorème très connu de Lagrange dit que cela est toujours possible. Ainsi notre équation devient :

$N = (1 + x_1^2 + x_2^2 + x_3^2 + x_4^2)(1 + y_1^2 + y_2^2 + y_3^2 + y_4^2)$.

Cela permet de coder la factorisation comme un MinRank sur $\mathbb{Z}$ avec des matrices de taille constante.

Méthode 2 On va utiliser le fait bien connu que pour factoriser N il suffit d'être capable d'extraire des racines carrés modulo N (ce qui est un cas particulier du problème MQ avec 1 équation et 1 variable dans $\mathbb{Z}/N\mathbb{Z}$). Pour cela il suffit de trouver $x \in \mathbb{Z}/N\mathbb{Z}$ tel que le rang de $M = \begin{pmatrix} x & a^2 \\ 1 & x \end{pmatrix}$ devienne égal à 1. C'est un MinRank$(2, 2, 2, 1, \mathbb{Z}/N\mathbb{Z})$.

Méthode 3 Comme MinRank sur un corps est un problème NP-complet, il va de soi que la factorisation peut également être écrite comme un MinRank sur un petit corps fini. Pour ce faire on peut d'abord coder la factorisation comme MQ, comme nous l'avons fait dans la section 10.4, puis re-coder MQ comme MinRank, comme dans le Corollaire 23.1.1.4.

23.2 MinRank diagonal, codes et lattices

MinRank reste également très difficile dans le cas ou toutes les matrices sont diagonales et $\eta = n$.

23.2.1 MinRank diagonal sur $\mathbb{Z}$

Quand les matrices sont diagonales, et sur $\mathbb{Z}$, le problème est de trouver une combinaison linéaire des vecteurs données qui a un petit nombre de coordonnées non-nulles. Ce problème est très voisin du célèbre problème de réduction de réseaux (lattice reduction problem) qui consiste à minimiser non pas le poids mais la longueur des vecteurs. Le problème de réduction des réseaux admet énormément d'applications en cryptographie. Dans le deux cas l'algorithm LLL risque de fonctionner jusqu'à environ $n = 300$. Par contre ce n'est probablement pas aussi simple pour le MinRank général, car on a affaire à un problème indécidable, voir Corollaire 23.1.1.2.

23.2.2 MinRank diagonal sur un corps fini

MinRank devient alors le problème de décodage du syndrome SD d'un code correcteur qui a beaucoup d'applications en cryptographie [127, 137, 139, 135, 134].

Autre Preuve que MinRank est NP-complet : On se ramène au problème SD qui est NP-complet. La preuve pour $q = 2$ se trouve dans [100], et la preuve dans le cas général est esquissée dans [108], page 1764. Soit un (n, k, d)-code linéaire G. Soit η polynomial en n. La réduction est triviale : Chaque ligne de la matrice génératrice de G sera la diagonale de la partie gauche $n \times n$ d'une matrice M_i $\eta \times n$, remplie avec des zéros partout ailleurs. De même M_0 contiendra le mot à décoder. Résoudre ce MinRank pour $r = d$ est alors équivalent à décoder G en corrigeant r erreurs. □

23.2.3 Le MinRank est-il un problème des codes correcteurs ?

Le MinRank général $\eta \times n$ est exactement le problème de décoder un sous-code d'un code correcteur de type rang sur $\mathbb{F}_{q^\eta}$, comme défini par Gabidulin [111], [1] , si l'on se restreint à une partie du code, celle qui est engendrée par des combinaisons linéaires dont les coefficients ne sont pas dans $\mathbb{F}_{q^\eta}$ mais dans $\mathbb{F}_q$. Il s'agit d'une classe de sous codes bien précise des codes de Gabidulin. Actuellement la meilleure attaque connue pour décoder les codes de type rang est justement basée sur MinRank, voir [110]. La meilleure attaque connue avant était l'énumération des bases de Stern-Chabaud [114].

Si q est premier, et dans ce cas seulement, MinRank est également, comme expliqué dans [110], le problème de décodage d'un "group code" [2] de longueur $n \times \eta$ sur $\mathbb{F}_{q^\eta}$ avec la métrique rang.

Comme nous l'avons déjà montré dans 23.2.2, MinRank contient également le problème SD, de décodage de codes correcteurs ordinaires (codes linéaires sur un corps).

23.3 Conclusions

23.3.1 MinRank est-il exponentiel ?

SD un problème NP-complet bien étudié [100, 108, 102, 105]. Le problème se pose constamment en théorie des codes depuis une trentaine d'années, et pourtant, assez souvent, les codes correcteurs bien connus et étudiées, ont leur poids minimal qui reste inconnu. Le problème semble dur non seulement dans le pire de cas, mais aussi en moyenne. Les meilleurs algorithmes connus pour résoudre ce problème s'avèrent exponentiels aussi bien en théorie qu'en pratique, bien qu'ils soient meilleurs que la recherche exhaustive. Tous ces algorithmes sont étudiés avec soin dans [101], et aussi dans [139] (moins récent). On peut aussi consulter [100, 108, 105, 102, 103, 106].

MinRank généralise SD, il y a donc de raisons d'espérer que MinRank soit au moins aussi dur. Dans la partie qui suit on va étudier plusieurs algorithmes pour résoudre MinRank. Ils s'avèrent tous exponentiels.

[1] On appelle ces codes également les RD-codes, de l'anglais *Rank Distance*

[2] Un "group code" de longueur N sur $\mathbb{F}_q$ est un sous-groupe additif de $\mathbb{F}_q^N$ avec l'addition composant par composant. Cette notion généralise des codes linéaires sur les anneaux, qui à leur tour généralisent des codes linéaires sur des corps finis qui sont les plus utilisés.

Chapitre 24

Les attaques connues pour MinRank

Soit $2 \leq \omega \leq 3$ l'exposant de la réduction de Gauss.

24.1 Les calculs de probabilités

Suivant les calculs classiques, voir [111], le nombre de matrices $\eta \times r$ de rang r est égal à :

$$A_r(r) = (q^\eta - 1) \cdot \ldots \cdot (q^\eta - q^{r-1}).$$

On définit suivant [111] :

$$\begin{bmatrix} n \\ r \end{bmatrix} = \frac{(q^n - 1) \cdot \ldots \cdot (q^n - q^{r-1})}{(q^r - 1) \cdot \ldots \cdot (q^r - q^{r-1})}$$

On peut alors calculer, toujours suivant [111] ou [110], le nombre exact de toutes les matrices $\eta \times n$ de rang r :

$$A_r(n) = \begin{bmatrix} n \\ r \end{bmatrix} \cdot A_r(r).$$

La probabilité qu'une matrice $\eta \times n$ choisie au hasard soit de rang r est donnée par :

$$P(\eta, n, r) = \frac{A_r(n)}{q^{\eta n}}.$$

Si $r \leq min(n, \eta) = n$, cette probabilité est différente de zéro et pour $n, \eta, r \rightarrow \infty$ on obtient assez facilement que :

$$P(\eta, n, r) = \frac{\left[\begin{smallmatrix} n \\ r \end{smallmatrix}\right] \cdot A_r(r)}{q^{\eta n}} = \mathcal{O}(q^{(\eta+n)r - r^2 - \eta n})$$

24.2 Evaluation de m maximum

Suivant le calcul de 24.1 la probabilité qu'une matrice aléatoire $\eta \times n$ ait un rang r est égale à $\mathcal{O}(q^{(\eta+n)r-\eta n-r^2})$.

On aura besoin de

$$m_{max} \stackrel{def}{=} \eta n + r^2 - (\eta + n)r + 1$$

matrices afin d'avoir en moyenne 1 solution au MinRank. En effet, le nombre de combinaisons linéaires des M_i en tenant compte des matrices co-linéaires sur $\mathbb{F}_q$, est

$$(q^{m_{max}} - 1)/(q - 1) = \mathcal{O}(q^{m_{max}-1})$$

et cela doit être égal approximativement à

$$\mathcal{O}(q^{\eta n+r^2-(\eta+n)r}).$$

Maintenant, si $m > m_{max}$, on peut fixer à des valeurs arbitraires les $m - m_{max}$ parmi les α_i, et on va toujours avoir une solution en moyenne. Ainsi si $m > m_{max}$ on a réduit le problème $MinRank(\eta, n, m, r, q)$ à $MinRank(\eta, n, m_{max}, r, q)$.

Conclusion : Avant d'appliquer tout algorithme à $MinRank(\eta, n, m, r, q)$, on pose

$$m := Min(m,\ \eta n + r^2 - (\eta + n)r + 1).$$

Important : Cette remarque s'applique à une instance aléatoire du MinRank. Cela ne s'applique pas forcément à des instances générés à partir d'autres problèmes NP-complets comme décrit dans 23. Dans ces cas particuliers, il peut être nécessaire d'évaluer séparément la loi de probabilités que suivent les rangs.

Remarque sur la symétrie du problème

Dans les attaques qui suivent dans les sections 24.4, 24.5 et 24.6, on peut toujours transposer toutes les matrices, ce qui revient à échanger les rôles de η et n. Fort heureusement, avec l'hypothèse habituelle $\eta \geq n$, il s'avère que les complexités des versions transposées sont systématiquement plus grandes. Ainsi on obtiendra directement les versions optimales des attaques par rapport à la transposition.

24.3 Attaques par énumération

La recherche exhaustive est en

$$q^m \cdot r^\omega.$$

On remarque que le facteur polynomial r^ω donné ne contient pas ni η ni n. En effet dans la recherche exhaustive il existe une stratégie de "early abort". La plupart du temps on va vérifier le rang d'une sous-matrice $(r+1) \times (r+1)$ et constater qu'il est $> r$. Comme la plupart du temps on tombera sur des sous-matrices $(r+1) \times (r+1)$ de rang maximal, la vérification complète sera faite rarement et le surcoût occasionné sera amorti. De même le coût de construction d'une matrice - combinaison aléatoire de m matrices M_i sera amorti, il suffit d'additionner des combinaisons linéaires déjà construites. Ainsi le coût moyen de construction sera seulement $\mathcal{O}(r^2)$ au lieu de $m \times r^2$ et $\mathcal{O}(r^2) \leq r^\omega$.

La recherche exhaustive est loin d'être une attaque inutile. Il arrive en effet que m devienne nettement plus petit que la valeur initiale, grâce à la substitution $m = m_{max}$ (voir 24.2). Ainsi les attaques exhaustifs peuvent s'avérer redoutables dans certains cas. Par example :

24.3.1 Attaque pour MinRank $n \times n$ avec $r \approx n$.

Soit $\eta = n$ et $r = n - s$. On a :

$$m_{max} = r^2 + n^2 - 2nr = s^2.$$

Donc casser un MinRank $n \times n$ avec $r = n - s$ nécessite au plus $q^{s^2} \cdot n^3$ opérations. Par example pour $r = n - 2$ on a une attaques en $q^4 \cdot n^3$. Ce fait m'a été suggéré par prof. Claus P. Schnorr.
Cela ne veut pas dire que le MinRank avec le rang proche du maximum n'est jamais solide. En effet, il suffit que les matrices ne soient pas quadratiques, pour éviter ce type d'attaque. Par example, avec $n \times 2n$ matrices et $r = n - 1$, il est facile à voir que $m_{max} = n + 1$ et l'attaque redevient exponentielle.

24.4 Attaques pour MinRank avec $m >> n$

24.4.1 L'algorithme "big m"

La première remarque est que si $m > \eta(n - r)$, MinRank est facile de résoudre par la réduction de Gauss. Il suffit d'obtenir des zéros dans les $n - r$ colonnes de la matrice M à trouver. Sinon, pour $m \leq \eta(n - r)$ l'attaque reste toujours possible dans une certaine mesure. La probabilité qu'elle fonctionne est celle que les $\eta(n - r) - m + 1$ éléments restants soient également des zéros.

La complexité de cette attaque est donc

$$q^{Max(0,\eta(n-r)-m+1)}\,(\eta(n-r))^{\omega}$$

24.4.2 L'algorithme du syndrome

Cette attaque est toute récente (mars 2001) et développe une remarque faite par prof. Ernst M. Gabidulin.
On va supposer $r < n$. Au lieu d'écrire l'espace affine des matrices comme un image d'une application linéaire ou affine, (le paradigme de la matrice génératrice d'un code), on va l'écrire comme un noyau (approche de type syndrome). Soit $\mathcal{S} : GF(q)^{\eta n} \to GF(q)^{\eta n - m}$ la partie linéaire des $\eta n - m$ équations affines du syndrome. Soit σ la partie constante. Soit

$$A \stackrel{def}{=} Min\left(\frac{\eta n - m - 1}{2}, \eta n + r^2/4 - (\eta + n)r/2 - 1\right)$$

L'attaque fonctionne comme suit :

1. Générer q^A matrices non-colinéaires $\eta \times n$ de rang $\leq r/2$.
   Par définition $A \leq \eta n + r^2/4 - (\eta + n)r/2 - 1$, et les calculs de cardinalité faits dans 24.1 montrent que cela est précisément la condition pour qu'il existe suffisamment de matrices non-colinéaires de rang r.
   Cette étape prend au plus $q^A \cdot \mathcal{O}(r\eta n)$ pas de calcul.
2. On va fabriquer deux listes, une avec les syndromes complets $\mathcal{S}(X_i) + \sigma$ et l'autre avec des syndromes sans la partie constante $\mathcal{S}(X_j)$.
3. Trouver une collision sur les premières $2A$ positions du syndrome sur le total de $(\eta n - m)$.
4. Le rang de la différence $M = X_{-}X_j$ des 2 matrices correspondantes est $\leq r$.
   Le syndrome de M est correct et égal à σ sur les premières $2A$ positions.
5. Les cardinal des paires différentes est environ q^{2A}. Le nombre de répétitions nécessaires pour que tout le syndrome soit correct est environ $q^{\eta n - m - 1}/q^{2A} \geq 1$

La complexité de l'attaque est :

$$q^{\eta n - m - 1}/q^{2A} \cdot q^A \cdot \mathcal{O}(r\eta n)$$

$$q^{\eta n - m - 1 - Min\left(\frac{\eta n - m - 1}{2}, \eta n + r^2/4 - (\eta + n)r/2 - 1\right)} \cdot \mathcal{O}(r\eta n)$$

$$q^{Max\left(\frac{\eta n - m - 1}{2}, (\eta + n)r/2 - m - r^2/4\right)} \cdot \mathcal{O}(r\eta n)$$

24.5 Algorithmes pour MinRank avec $r << n$

Les algorithmes présentées dans cette partie ont en commun de fonctionner uniquement si r est petit.

24.5.1 La réduction de MinRank à MQ [Shamir]

Cet algorithme réduit MinRank à MQ et avait été décrit pour la première fois par Shamir et Kipnis dans [71]. Nous en donnons une variante qui est plus claire et plus simple à analyser.

Soit M la matrice recherchée de rang r. Au début chacune des entrées de M est connue en tant qu'une expression affine :

$$M_{ij} = M_{0ij} + \sum_k \alpha_k M_{kij}$$

Soit L_i la i-ème ligne de M écrite en tant que η expressions linéaires (affines) en les α_j. Le rang de M est r et on peut supposer que les r premières lignes donnent déjà le rang r, sinon l'attaque va échouer et il faudra recommencer avec un autre sous-ensemble de lignes après permutation.
Pour exprimer MinRank comme MQ, on va écrire des équations sur des lignes L_i de la matrice recherchée M qui disent que chacune des lignes $L_{r+1}, \ldots, L_n]$, est une combinaison linéaire des lignes précédentes $L_1, \ldots, L_r$.

$$L_{r+1} = \sum_{i=1}^{r} \beta_{ir} L_i$$

$$\vdots$$

$$L_n = \sum_{i=1}^{r} \beta_{ir} L_i$$

Chaque de ces équations sur des lignes est constituée de η équations sur les α_i. En tout, pour les lignes $L_{r+1}, \ldots L_n$, on obtient $\eta(n-r)$ équations sur les α_i. Le nombre de variables est m pour les α_i, plus $r(n-r)$ pour les nouvelles variables $\beta_{ij}, i \geq r+1$.
Ainsi on a pu écrire la condition de MinRank comme $\eta(n-r)$ équations quadratiques avec $r(n-r)+m$ variables sur K. Ce système est sur-défini (*overdefined* en anglais). Si l'on note le nombre de variables

$$n'' = r(n-r) + m,$$

le nombre d'équations est :

$$m'' \; = \; \eta(n-r) \; \approx \; \frac{1}{r^2} \frac{\eta}{n-r} n''^2$$

Soit

$$\varepsilon \approx \frac{\eta(n-r)}{[r(n-r)+m]^2}.$$

On va considérer que nous avons un système de $m'' = \varepsilon n''^2$ équations quadratiques avec n'' variables. Dans 7.5.2, la complexité de l'attaque XL sur un tel problème est estimée, avec en supposant que ε est constant.

$$(n'')^{\frac{\omega}{\sqrt{\varepsilon}}} / \sqrt{\frac{1}{\varepsilon}}!$$

On va aller au delà de l'hypothèse sur ε et appliquer la formule de 7.5.2, pour avoir une estimation de faisabilité de l'attaque. On va aussi supposer que $m \ll r(n-r)$ qui est toujours les cas dans tous les examples étudiés dans ce manuscript. Cela donne :

$$\varepsilon \approx \frac{\eta}{r^2(n-r)}$$

$$(n'')^{\frac{\omega}{\sqrt{\varepsilon}}} / \sqrt{\frac{1}{\varepsilon}}! \;\approx\; (r(n-r))^{\omega r \sqrt{(n-r)/\eta}} / (r\sqrt{(n-r)/\eta})! \;\approx\; (nr)^{\sqrt{\frac{n}{\eta}}\omega r} / r! \;\approx$$

$$\approx\; C \cdot n^{\omega r}$$

Il est important de comprendre que l'attaque (et l'estimation) n'est applicable **uniquement** si $r << n$. En effet, seulement dans le cas $r << n$, il est vrai que $m'' >> n''$ et le système MQ est sur-défini. Or l'attaque XL et sa complexité donnée dans 7.5.2 s'applique uniquement dans ce cas.

24.5.2 La méthode avec des sous-matrices

Cette attaque marche également **uniquement** pour $r << m$. L'idée est très simple et avait déjà été utilisée par Coppersmith, Stern et Vaudenay dans [77], page 439 et ensuite dans [78]. Elle exprime directement MinRank par des déterminants et nous permet dans 13 et [68] d'améliorer considérablement l'attaque de Shamir-Kipnis sur HFE.

On va encore écrire la matrice M à trouver avec des entrées étant des expressions linéaires (affines) en les α_j. Pour toute sous matrice $(r+1)$x$(r+1)$ N de M on écrit :

$$\forall_N \begin{cases} 0 = \text{dét}(N) \\ \vdots \end{cases}$$

On peut écrire $\binom{\eta}{r+1} \cdot \binom{n}{r+1}$ de telles équations. ce sont des équations de degré au plus $r+1$ en les α_i et donc le nombre de termes $\alpha_i\alpha_j\alpha_k$.. qui y apparaissent est de l'ordre de

$$\sum_{i=0}^{r+1} \binom{m}{i}$$

Pour que l'attaque marche il faut que :

$$\sum_{i=0}^{r+1} \binom{m}{i} \leq \binom{\eta}{r+1} \cdot \binom{n}{r+1} \tag{24.1}$$

A ce moment là, elle aura pour complexité

$$(\sum_{i=0}^{r+1} \binom{m}{i})^{\omega}$$

L'attaque n'est faisable **que** si l'on a $r << m$, car sinon $\sum_{i=0}^{r+1} \binom{m}{i} \to 2^m$ et serait beaucoup trop grand.

Avec $r << m$ la complexité de l'attaque est environ

$$(\binom{m}{r+1})^3 \approx m^{\omega(r+1)}/(r+1)!$$

Il convient d'insister sur le fait que cette complexité ne s'applique **que** si r est suffisamment petit pour que l'on ait l'inégalité 24.1.

24.6 L'attaque du noyau

Cette attaque est la plus puissante de toutes les attaques actuellement connues pour MinRank. Même si elle est conçue à l'origine pour r petit, elle est marchera toujours avec la complexité annoncée, et s'avère être la meilleure attaque connue bien au delà de cette contrainte. Elle avait été inventée par Louis Goubin dans [80] pour cryptanalyser le cryptosystème TTM [79]. Nous allons décrire une variante de cette attaque plus générale que celle donnée dans [80]. D'autres versions et améliorations de cette attaque sont décrites dans [110].

La version de base de l'attaque du noyau

L'attaque consiste à deviner un certain nombre de vecteurs qui appartiennent au noyau de la matrice recherchée M. Puisque le rang de M est r, la probabilité qu'un vecteur soit dans le noyau gauche est q^{-r}. On va essayer de deviner $\lceil \frac{m}{\eta} \rceil$ vecteurs qui sont dans le noyau gauche de M. Avec la probabilité de succès de q^{-r} pour chacune d'entre eux, en $q^{\lceil \frac{m}{\eta} \rceil r}$ on arrive à deviner $\lceil \frac{m}{\eta} \rceil$ vecteurs du noyau gauche $X_1, \ldots, X_{\lceil \frac{m}{\eta} \rceil}$. Ensuite, chacun de ces vecteurs donnera η équations linéaires sur α :

$$0 \;=\; M \cdot X_i \;=\; \sum_{j=1}^{m} \alpha_i (M_j \cdot X_i) - M_0 \cdot X_i$$

En tout on a $\lceil \frac{m}{\eta} \rceil \cdot \eta \geq m$ équations linéaires à résoudre, avec m inconnues α_i. La complexité de l'ensemble de l'attaque est :

$$q^{\lceil \frac{m}{\eta} \rceil r} \cdot m^{\omega}$$

Version étendue de l'attaque du noyau

Dans certain cas, quand m/η ne dépasse que de très peu un nombre entier, il est intéressant de se restreindre à un espace linéaire de dimension m', m' étant un multiple de η proche de m défini par $m' = \lfloor \frac{m}{\eta} \rfloor n$. La probabilité que la solution soit dans l'espace sélectionné est $q^{m'-m} = (m \bmod \eta)$. Dans l'ensemble, la version modifiée de l'attaque a pour complexité :

$$q^{(m \bmod \eta)} \cdot q^{\lfloor \frac{m}{\eta} \rfloor r} \cdot m^{\omega}$$

La complexité de la meilleure version de l'attaque du noyau est en général donnée par :

$$Min\left(\quad q^{\lceil \frac{m}{\eta} \rceil r} \quad , \quad q^{\lfloor \frac{m}{\eta} \rfloor r + (m \bmod \eta)} \quad \right) \cdot m^{\omega}$$

24.7 Exemples

Le tableau suivant resume la complexité de tous les attaques connus pour le problème MinRank. Pour chacune des instance du problème on donne la probabilité d'avoir une solution calculée avec l'aide de la formule donnée dans 24.1, ainsi que la complexité de toutes les attaques présentées.
L'évaluation avait été faite pour 6 examples de paramètres $A - F$ pour le schéma d'authentification MinRank introduit dans 25.2, et avec deux examples de MinRank qui apparaît dans l'attaque de Shamir-Kipnis sur HFE 13.2.

Cryptosystem	MinRank identification						HFE	
Parameter set	A	B	C	D	E	F	Chall. 1	Quartz
m	10	10	10	81	121	190	80	103
n	6	7	11	19	21	29	80	103
η	6	7	11	19	21	29	80	103
r	3	4	8	10	10	15	7	8
q	65521	65521	65521	2	2	2	2^{80}	2^{103}
$Pr_\alpha[Rank \leq r]$	0.6	0.6	0.6	0.6	0.6	2^{-6}	$< 2^{-10^5}$	
20×Comm. [Kb]	1.94	2.99	4.86	2.17	2.36	3.13		
Attack								
Brute force	2^{168}	2^{168}	2^{170}	2^{81}	2^{134}	2^{205}	2^{80}	2^{103}
Kernel	$\mathbf{2^{106}}$	$\mathbf{2^{122}}$	$\mathbf{2^{138}}$	$\mathbf{2^{64}}$	$\mathbf{2^{81}}$	$\mathbf{2^{128}}$	2^{577}	2^{844}
Big m	2^{108}	2^{205}	2^{399}	2^{113}	2^{135}	2^{243}	2^{461k}	2^{997k}
Syndrome	2^{118}	2^{312}	2^{1002}	2^{151}	2^{172}	2^{339}	2^{252k}	2^{530k}
Sub-matrices	∞	∞	∞	∞	∞	∞	$\mathbf{2^{97}}$	$\mathbf{2^{114}}$
MQ	∞	∞	∞	∞	∞	∞	2^{152}	2^{188}

Table 24.7.1. L'application des attaques connues sur MinRank

Les examples A-F concernent notre schéma MinRank de la section 22.1 et ont été sélectionnés de sorte à obtenir, pour des différents niveaux de sécurité allant de 2^{64} à 2^{128}, la meilleure complexité en communication "20×comm. [Kb]", pour la meilleure variante en 20 rounds du schéma de 22.1.
Le premier example pour HFE est le "HFE Challenge 1" décrit dans 13.1 et [65, 70, 68]. Le deuxième est un sous-ensemble strict de Quartz décrit dans [66, 67] et mentionné dans 18.4.3. Cette attaque en 2^{114} ne permet pourtant pas de cryptanalyser Quartz tout entier.

Chapitre 25

MinRank en authentification

Dans cette partie nous allons introduire un nouvel algorithme asymétrique. C'est un algorithme d'authentification à divulgation nulle de connaissance (Zéro-knowledge) basé sur le problème NP-complet MinRank. Il est basé sur le calcul matriciel avec des matrices $\eta \times n$ sur un corps fini K.

Le principe de l'authentification avec MinRank est le suivant : il est facile de camoufler complètement une matrice M de rang $r < n$ en faisant deux transformations inversibles S et T :

$$M' = T \cdot M \cdot S.$$

La distribution de probabilités de M' est celle d'une matrice aléatoire de rang r, comme montré dans le Lemme 25.1.0.1.

Supposons que l'on veuille prouver (ou convaincre) que

$$M = \sum_i \alpha_i M_i$$

sans révéler α. Pour cela on va choisir S et T, et mettre $M' = T \cdot M \cdot S$ en gage. Ensuite on va ou bien

1. révéler M' et l'adversaire vérifiera que M' est bien de rang r,
2. ou alors on va donner S et T et prouver que M' s'écrit comme une combinaison linéaire des TM_iS.

Il reste à faire cette dernière preuve (2) en Zéro-knowledge, c'est à dire sans révéler α, et cela est possible, comme on le verra par la suite, en divisant simplement M' en deux parts P_1 et P_2 dont chacune séparément est aléatoire, et telles que $P_2 - P_1 = M'$.

25.1 Préliminaires

Théorème 25.1.0.1 *Soit M une matrice $\eta \times n$ de rang r. Soient S et T deux matrices $\eta \times \eta$ et $n \times n$ uniformément distribuées. Alors TMS est uniformément distribué parmi toutes les matrices $\eta \times n$ de rang r.*

Preuve : Toutes les matrices $\eta \times n$ de même rang r sont équivalentes [1] et peuvent être écrites comme :

$$M = S' \cdot \begin{pmatrix} Id_{r\times r} & 0_{r\times(n-r)} \\ 0_{(\eta-r)\times r} & 0_{(\eta-r)\times(n-r)} \end{pmatrix} \cdot T'$$

Le nombre de façons de passer d'une matrice P à une autre matrice Q est toujours le même. En effet, l'ensemble de ces transformation est en bijection évidente avec l'ensemble des changements de variables (S, T) qui transforment la matrice suivante en elle même :

$$\begin{pmatrix} Id_{r\times r} & 0_{r\times(n-r)} \\ 0_{(\eta-r)\times r} & 0_{(\eta-r)\times(n-r)} \end{pmatrix}. \quad \square$$

25.1.1 L'initialisation du schéma

La clef publique : Elle est composée de $1+m$ matrices non-singulières $\eta \times n$ sur un corps fini $GF(q)$,

$$M_0;\ M_1, \ldots, M_m.$$

Soit $r < n$. Afin de générer une instance aléatoire du problème MinRank ayant une solution (construite), on va choisir $1+(m-1) = m$ matrices inversibles [pseudo-]aléatoires $M_0;\ M_1, \ldots, M_{m-1}$. Ensuite on choisit M encore [pseudo-]aléatoire et de rank r, et ensuite on "adapte" M_m. Pour cela on choisi un $\alpha \in GF(q)^m$ complètement aléatoire, sauf que $\alpha_m \neq 0$, et M_m est obtenue comme :

$$M_m = (M + M_0 - \sum \alpha_i M_i)/\alpha_m.$$

Afin de réduire la taille de la clef publique, la plupart d'entre elles ne sont pas transmises, mais générées avec un générateur pseudo aléatoire déterministe. Les M et $M_1, \ldots, M_{m-1}$ seront générées à partir d'une graine de 160 bits. Il est préférable de prendre tous les M_i inversibles et de demander au Vérifieur de le vérifier, mais cela n'est pas nécessaire en pratique si la personne qui génère les clefs est honnête.

[1] Deux matrices M, M sont justement équivalentes s'il existe deux matrices inversibles de tailles appropriées avec $M' = TMS$. En fait deux matrices sont équivalentes si et seulement si elles ont le même rang.

Pour générer une matrice non-invertible de façon déterministe à partir d'un générateur pseudo-aléatoire on peut utiliser la très utilisée méthode LU. Quant à la génération de M, il faut d'abord générer une matrice L qui est composée d'une matrice pseudo-aléatoire et inversible $r \times r$ complétée avec les 0's à une matrice de taille $\eta \times n$. Ensuite on applique un paire de matrices pseudo-aléatoire inversibles S et T. Ainsi on prend $M = SLT$, voir le Lemme 25.1.0.1.

La clef secrète : C'est le $\alpha \in GF(q)^n$ que nous avons choisi de sorte à avoir : $Rank(M = \sum \alpha_i \cdot M_i - M_0) = r$.

Tailles de clefs : Toutes les matrices sauf une sont générées à partir d'une graine de 160 bits. La taille de la clef publique est donc $160 + \eta n \log_2 q$ bits. La clef nécessite $m \log_2 q$ bits pour écrire α, plus la taille de la clef publique qui ne peut pas être dérivée à partir de la clef secrète seule.

25.2 Le schéma d'identification MinRank

Dans le schéma qui suit, le Prouveur va convaincre le Vérifieur qu'il connaît α (et M). Chaque étape d'algorithme d'authentification, sur 20 ou 35 à prévoir, est identique et indépendante.
Soit H une fonction à sens unique à collisions faibles difficiles (Second Pre-image Resistant ou SPR, voir **??**. On suppose que cette fonction se comporte comme un oracle aléatoire. En pratique on utilisera une fonction de hachage connue comme SHA-1 dont en prendra le nombre bits de sortie nécessaires.

Une étape d'authentification

Le Prouveur choisit deux matrices inversibles S, T qui sont respectivement $\eta \times \eta$ et $n \times n$. Il prend aussi une matrice $\eta \times n$ dite "de camouflage" X qui est totalement aléatoire. On appelle $\overline{STX}$ le triplet (S, T, X). Ensuite il choisit une combinaison aléatoire β_1 des M_i :

$$N_1 = \sum \beta_{1i} \cdot M_i$$

Ensuite le Prouveur pose $N_2 = M + M_0 + N_1$ et utilise son expression secrète de M pour obtenir :

$$N_2 = \sum \beta_{2i} \cdot M_i$$

On a $\beta_2 - \beta_1 = \alpha$, mais chacune des β_i (prise séparément) est parfaitement aléatoire et équidistribuée. De même chacune des N_i a la distribution des probabilités d'une combinaison des M_i parfaitement aléatoire. En même temps on a

$$N_2 - N_1 = M + M_0.$$

1. Le Prouveur envoie au Vérifieur :

$$\xrightarrow{\hspace{4cm}}$$

$$H(\overline{STX}),\ H(TN_1S+X),\ H(TN_2S+X-TM_0S)$$

2. Le Vérifieur choisi une question $\mathcal{Q} \in \{0,1,2\}$ et envoie $\mathcal{Q}$ à P.

$$\xleftarrow{\hspace{4cm}}$$

$$\mathcal{Q} \in \{0,1,2\}$$

3. Si $\mathcal{Q} = 0$, le Prouveur va révéler les valeurs suivantes :

$$\xrightarrow{\hspace{4cm}}$$

$$(TN_1S+X),\ (TN_2S+X-TM_0S)$$

Vérification $\mathcal{Q} = \mathbf{0}$: Le Vérifieur va accepter si $H(TN_1S+X)$ et $H(TN_2S+X-TM_0S)$ sont corrects, et si

$$(TN_2S+X-TM_0S)-(TN_1S+X)=TMS$$

est en effet une matrice de rang r.

3' Dans les cas $\mathcal{Q} = 1,2$, le Prouveur va révéler :

$$\xrightarrow{\hspace{4cm}}$$

$$\overline{STX},\ \beta_{\mathcal{Q}}$$

Vérification $\mathcal{Q} = \mathbf{1},\mathbf{2}$: Le Vérifieur vérifie si S et T sont inversibles et si $H(\overline{STX})$ est correct. Ensuite il calcule

$$TN_{\mathcal{Q}}S = \sum \beta_{\mathcal{Q}i}\ \ TM_iS$$

et vérifie la valeur de $H(TN_1S+X)$ ou $H(TN_2S+X-TM_0S)$.

25.3 Les propriétés exigées

Un schéma d'authentification à divulgation nulle de connaissance (Zéro-knowledge), comme défini dans [145], doit satisfaire les 3 conditions classiques : Il faut montrer que le schéma est **consistant** (en anglais : completeness), **significatif** (en anglais : soundness) et **Zéro-knowledge**. Pour plus de détails voir l'abondante littérature à ce sujet, par example [3, 5, 155, 157, 158, 133]. Pour avoir ces notions en français, voir [150] ou la traduction française de [5] par Serge Vaudenay.

25.3.1 Consistance

Il est évident qu'un Prouveur légitime peut **toujours** répondre aux toutes les questions.

25.3.2 Significativité

On va montrer qu'un faux Prouveur n'est **jamais** en mesure de répondre à chacune de questions possibles $\mathcal{Q} = 0, 1, 2$. Soit C (Charlie ou le triCheur)), soit une machine de Turing probabiliste polynomiale. On va montrer qu'un Prouveur qui est capable de répondre à chacune des questions possibles connaît en fait le secret α (ou un équivalent si le MinRank a plusieurs solutions).

Tout d'abord C s'engage (avec H) sur les valeurs de $TN_1S + X$ et $TN_2S - TM_0S + X$. Pour $\mathcal{Q} = 1$ et 2 il va démontrer qu'il les a généré effectivement sous la forme $X + T(\sum \beta_{1i}M_i)S$ et $X + T(\sum \beta_{2i}M_i)S - TM_0S$. Dans les deux cas on vérifie $H(\overline{STX})$ et on sait qu'il doit utiliser les mêmes X, S et T pour répondre aux deux cas.

Enfin, pour $\mathcal{Q} = 0$ on va vérifier que le rang de la matrice suivante est réellement r :

$$\Big(T(\sum \beta_{2i}M_i)S - TM_0S + X\Big) - \Big(T(\sum \beta_{1i}M_i)S + X\Big) =$$

$$= \sum_{i=1}^{m}(\beta_{2i} - \beta_{1i}) \cdot TM_iS - TM_0S$$

Puisque pour $\mathcal{Q} = 1$ ou 2 on vérifie bel et bien que S et T sont inversibles, donc

$$\sum_{i=1}^{m}(\beta_{2i} - \beta_{1i}) \cdot M_i - M_0$$

est également de rang r. Ainsi le Prouveur connaît une solution au MinRank qui est précisément $(\beta_2 - \beta_1)$. Il connaît la clef secrète ou une quantité équivalente.

Il est facile à voir que la probabilité de fraude sur plusieurs itérations de l'algorithme est :

$$Pr_{fraud} = \left(\frac{2}{3}\right)^{\#\text{itérations}}.$$

Dans 25.5 on montre une astuce pour à réduire ce nombre à $\left(\frac{1}{2}\right)^{\#\text{itérations}}$. A l'occasion, dans 25.5.3 on explicite tous les scénarios de fraude possibles.

25.4 Le preuve que MinRank est Zéro-knowledge

Soit le Prouveur une machine Turing probabiliste polynomiale. On supposera que H se comporte comme une fonction aléatoire (ou un oracle aléatoire). Le simplicité du MinRank fait qu'il est facile de voir qu'il est Zéro-knowledge.

- Dans les cas $\mathcal{Q} = 1, 2$ on révèle uniquement des variables aléatoires indépendantes $S,\ T,\ \beta_{\mathcal{Q}}, X$.
- Le cas $\mathcal{Q} = 0$: révéler (TN_1S+X) et $(TN_2S+X-TM_0S)$ revient à révéler le premier (TN_1S+X) et leur différence $TN_2S-TM_0S-TN_1S = TMS$. Comme X est aléatoire équidistribuée indépendante de S, T, M, $(TN_1S + X)$ est aussi aléatoire équidistribuée indépendante de TMS. Quant à TMS, le Lemme 25.1.0.1 montre que c'est une matrice aléatoire équidistribuée parmi toutes les matrices de rang r.

Cela ne suffit pas, le Zéro-knowledge n'est pas seulement le fait qu'un adversaire (passif) n'apprend rien. Une définition moderne de Zéro-knowledge demande qu'un adversaire actif n'apprenne rien, tout en essayant d'extraire de l'information avec une stratégie quelconque. Cette notion est définie dans [145], voir aussi par example [140, 141, 142, 146]. Le schéma doit être sûr **quelque soit** l'adversaire, pas seulement dans le cas moyen ou pour la plupart des adversaires. La définition sera basée sur la notion de simulation. On va utiliser l'adversaire comme une boîte noire pour construire un simulateur de tout ce qu'un adversaire peut enregistrer au cours de son interaction avec le Prouveur légitime. Pour cette raison on parle "black-box Zero-knowledge". C'est une notion très forte et pourtant réaliste. On montre que l'adversaire n'a pu extraire aucune information du Prouveur, car il peut obtenir la même distribution des probabilités de ses enregistrements, sans même avoir accès au Prouveur. Ainsi, on peut se passer du Prouveur. Seule l'interaction en temps réel avec le Prouveur permet au Vérifieur d'être convaincu qu'il parle au Prouveur légitime, avec notamment un libre choix de questions. Sinon, le Vérifieur ne peut à posteriori se convaincre lui-même. Rien ne prouve que les questions posées ont été choisies au hasard. La simulation va précisément anticiper certaines questions qui seront posées et demandera plusieurs tentatives (re-winding) pour tomber juste. On va rappeler la définition complète du Zéro-knowledge, d'après Goldreich et Oren [145].

Definition 25.4.0.1 (Black box Zero-knowledge) *Une strategie P est dite "black box Zero-knowledge" pour une entrée commune $x \in S$ (la distribution des clefs publiques) s'il existe un algorithme de simulation efficace U tel que pour toute strategie faisable du Vérifieur V, les deux distributions de probabilités suivantes sont calculatoirement indistinguables :*

- $\{(P,V)(x)\}_{x \in S} \stackrel{def}{=}$ *toutes les sorties de V, quand il réagit avec P avec l'information commune $x \in S$.*
- $\{U(V)(x)\}_{x \in S} \stackrel{def}{=}$ *les sorties de U sur $x \in S$ et en utilisant V comme un oracle.*

25.4.1 Le preuve complète du Zéro-knowledge

On définit un simulateur U qui a accès à V en tant que boîte noire (oracle).

1. $U(V)$ choisi une question aléatoire $\mathcal{Q} = 1, 2$. Il va se préparer à répondre aux questions 0 **and** $\mathcal{Q}$.
2. Il choisit N sous forme $N = \sum \delta_i M_i$ avec un δ aléatoire.
3. Il choisit $\overline{STX} = (S, T, X)$ avec S et T inversibles.
4. Il choisit une matrice aléatoire R de rang r.
5. Soient $N_{\mathcal{Q}} = N$ et $N_{3-\mathcal{Q}} = N + (-1)^{\mathcal{Q}+1}(R + M_0)$. Maintenant $N_2 - N_1 = R + M_0$.
6. Le simulateur va demander au Vérifieur de poser une question sur ses valeurs mises en gage avec H :

$$\mathcal{Q}' = V\left(H(\overline{STX}),\ H(TN_1S + X),\ H(TN_2S - TM_0S + X)\right) \in \{0, 1, 2\}.$$

7. Le simulateur va répéter les étapes 1-6 environ 2 fois (**rewinding**), tant qu'il n'obtient pas une de deux questions qu'il est préparé à répondre :

$$\mathcal{Q}' \in \{0, \mathcal{Q}\}$$

8. Si $\mathcal{Q}' = 0$ le simulateur $U(V)$ révèle $(TN_2S + X - TM_0S)$ et $(TN_1S + X)$. Leur différence est

$$TN_2S + X - TM_0S - TN_1S - X = T(R + M_0)S - TM_0S = TRS$$

ce qui donne bel et bien une matrice de rang r.

8' Si $\mathcal{Q}' = \mathcal{Q}$ le simulateur $U(V)$ révèle $\overline{STX}$ et δ qui ont été en effet utilisés pour construire $TN_{\mathcal{Q}}S + X[-TM_0S]$.

25.5 Réduction de la probabilité de fraude

On présenter une astuce pour avoir une probabilité de fraude 1/2 au lieu de 2/3. Son principe est le suivant :

1. Remplacer certains choix aléatoires qui sont faits dans le protocole du côté Prouveur, de sorte qu'ils deviennent pseudo-aléatoires.
2. Il faut que la façon et l'ordre de les générer impose que certain scénarios de fraudes ne soient plus possibles.
3. Il faut également que le Vérifieur puisse vérifier que les valeurs ont bel et bien été générés de cette façon.

Par example, supposons que dans un protocole interactif, le Prouveur va choisir deux valeurs aléatoires β_1 et β_2 de sorte que $\beta_2 - \beta_1 = \alpha$ est une valeur secrète. Si on lui permet de choisir les β_i aléatoirement, ce qui n'est **pas vérifiable**, alors un fraudeur qui prend la place du Prouveur aura toujours la possibilité de choisir habilement au moins une de deux valeurs β_i en fonction de son but de fraude et selon les détails du schéma.

On peut toujours imposer que β_1 soit généré par un générateur pseudo-aléatoire, mais dans ce cas on ne peut pas éviter des fraudes dans lesquelles on choisit habilement β_2. Le problème de protéger les deux β_i de la manipulation paraît impossible à résoudre. Et pourtant il admet des solutions cryptographiques. On va en présenter ici une, basée sur le problème MQ (il en existe une analogue basée sur la factorisation).

Soit $F : GF(q)^{n\eta} \rightarrow GF(q)^{n\eta}$ un ensemble d'équations quadratiques aléatoires (MQ) qui est publique et dont les coefficients peuvent être générés par un générateur pseudo-aléatoire. On remarque que :

(1). Il est difficile de calculer un inverse $F^{-1}(y)$ pour un y aléatoire.

(2). Il est très facile de calculer deux solutions x et x' tels que $F(x') - F(x)$ est une valeur donnée Δy et tels que $x' = x + \Delta x$ avec une constante donnée Δx. En effet, $F(x + \Delta x) - F(x) = \Delta y$ est une équation linéaire.

25.5.1 Application au schéma MinRank

Dans notre modification du schéma décrit dans 25.2 le Prouveur va commencer par choisir deux graines aléatoires de 160-bits Z and $\overline{STX}$. Soient

$$\Delta y = \text{Expand}(Z)$$

$$(S, T, X) = \text{Expand}(\overline{STX})$$

avec un générateur pseudo-aléatoire Expand. Ensuite le Prouveur va résoudre :

$$\begin{cases} F(T(\sum \beta_{2i} M_i)S - TM_0S + X) & - & F(T(\sum \beta_{1i} M_i)S + X) & = & \text{Expand}(Z) \\ \beta_2 & - & \beta_1 & = & \alpha \end{cases}$$

La première équation devient linéaire en β_1 après la substitution de $\beta_2 = \beta_1 + \alpha$. On obtient m équations linéaires avec m variables β_{1i}. Si le système n'a pas de solution (β_1, β_2), on recommence un petit nombre de fois avec une autre graine Z.

25.5.2 Modifications dans les communications du schéma

Si $\mathcal{Q} = 0$, le Prouveur va envoyer au Vérifieur une valeur additionnelle Z pour la vérification.

25.5.3 Vérification que le Prouveur suit le scénario

Le Vérifieur va vérifier que

$$F(TN_2S - TM_0S + X) - F(TN_1S + X) = \text{Expand}(Z).$$

Dans la version précédente du schéma MinRank les scénarios de fraude possibles étaient :

01 Se préparer pour répondre à $\mathcal{Q} = 0$ et 1.
Pour frauder il suffit de fabriquer deux matrices, soi-disant $T(\sum \beta_{1i} M_i)S + X$ et $T(\sum \beta_{2i} M_i)S - TM_0S + X)$, telles qu'une seulement d'entre elles est construite sous cette forme, et l'autre est ajustée afin d'avoir leur différence de rang r.

02 Se préparer pour répondre à $\mathcal{Q} = 0$ et 2 de même façon.

12 Se préparer pour répondre à $\mathcal{Q} = 1$ et 2 :
Choisir au hasard $\overline{STX}, \beta_1, \beta_2$ et produire $T(\sum \beta_{\mathcal{Q}i} M_i)S + X[-TM_0S]$.

0 Se préparer pour répondre à $\mathcal{Q} = 0$ seulement. Pour cela on va juste produire n'importe quelles deux matrices telles que leurs différence est de rang r.

1 Se préparer pour répondre à $\mathcal{Q} = 1$ seulement. Pour cela on va produire $T(\sum \beta_{1i} M_i)S + X$ sous la forme souhaitée.

2 Se préparer pour répondre à $\mathcal{Q} = 2$ seulement. Comme plus haut.

La nouvelle version ne permet plus de faire les fraudes (01) et (02). Voyons en détails pourquoi sur l'example de (01). On suppose que le fraudeur souhaite répondre à $\mathcal{Q} = 0$ et 1. Il peut essayer un de scénarios suivants :

1. Puisque S, T et X sont toujours obtenus comme $\text{Expand}(\overline{STX})$, si on triche et on n e le choisit pas ainsi, il sera possible de répondre à la seule question $\mathcal{Q} = 0$.
2. Supposons qu'il commence par choisir β_1. Comme F est à sens unique (le problème MQ), il ne sera pas capable de produire une matrice Q telle que $F(Q) - F(T(\sum \beta_{1i} M_i)S + X) = \text{Expand}(Z)$.
3. Une autre possibilité serait de choisir R de rang r et écrire $n\eta$ équations avec m variables $(\sum \beta_{2i} M_i - M_0) - (\sum \beta_{1i} M_i) = R$. Or trouver une solution à ce système est difficile car $\alpha = \beta_2 - \beta_1$ donnerait une solution au problème difficile MinRank.

Par contre le fraudeur garde sa capacité à répondre à chacune des questions séparément : fraudes (0), (1) et (2). Les (1) et (2) sont triviales, puisque on n'a ajouté aucune vérification supplémentaire pour $\mathcal{Q} = 1, 2$.
Pour (0) c'est moins trivial. Le fraudeur doit présenter deux matrices J_1 et J_2 qui remplissent les deux conditions suivantes :

$$\left\{ \begin{array}{rcl} F(J_2) - F(J_1) & = & \text{Expand}(Z) \\ Rank(\ J_2 - J_1\ ) & = & r \end{array} \right. \tag{25.1}$$

Cela se fait en choisissant une matrice ΔJ de rang r, et en pose $J_2 = J_1 + \Delta J$ dans la première équation, qui devient alors linéaire on les J_{1ij}.

25.5.4 L'impact sur le schéma MinRank

Maintenant il est possible de modifier des probabilités des différentes question posées dans le schéma. La question $\mathcal{Q} = 0$ sera posée avec probabilité $1/2$ et les questions $\mathcal{Q} = 1, 2$ avec les probabilités de $1/4$. Les probabilités des réussite pour un adversaire qui essaie un des scénarios de fraude possible sont désormais :

scénario de fraude	0	1	2	01	02	12	012
Pr[Succès] avant	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0
après	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{2}$	0

Donc un fraudeur sera toujours détecté avec une probabilité de $1/2$. Ainsi plus que 20 itérations seulement, au lieu de 35, seront nécessaires pour avoir la sécurité de 10^{-6}. Cela aura pour effet de diminuer la complexité globale en communications calculée dans 26.1.

Remarque On obtient un schéma plus performant avec une hypothèse calculatoire supplémentaire basée sur la difficulté du problème NP-complet MQ. Si jamais ce problème s'avère moins solide que prévu, alors les scénarios (01) et (02) redeviendront possibles et la probabilité de fraude sera de $3/4$. Le schéma restera sûr, seulement la probabilité sera dégradée et il faudra d'avantage d'itérations.

25.6 Des modifications supplémentaires

Dans cette partie on va ajouter des modifications supplémentaires à la variante décrite dans le chapitre précédent 25.5, afin d'obtenir un schéma encore plus performant.

D'abord on remarque que si $\mathcal{Q} = 0$, il n'est pas du tout nécessaire de transmettre les deux valeurs $TN_2S - TM_0S + X$ et $TN_1S + X$. En fait il suffit de transmettre leur différence TMS et Z qui est déjà transmis dans la version modifiée. Les valeurs habituelles peuvent être récupérées par le Vérifieur en résolvant un système analogue à (25.1). On va économiser le transfert d'une matrice $\eta \times n$, ce qui permet de gagner en communications.

La deuxième amélioration consiste à n'utiliser qu'une seule graine appelée $\overline{STXZ}$ et poser :

$$(S, T, X, Z) = \text{Expand}(\overline{STXZ})$$

25.7 Le schéma modifié MinRank-v2

On va intégrer toutes les modifications pour avoir une vue d'ensemble. Le Prouveur va commence par choisir une graine aléatoire de 160-bits $\overline{STXZ}$. Soient

$$(S, T, X, Z) = \text{Expand}(\overline{STXZ})$$

$$\Delta y = \text{Expand}(Z)$$

Ensuite le Prouveur va résoudre :

$$\begin{cases} F(T(\sum \beta_{2i} M_i)S - TM_0S + X) & - & F(T(\sum \beta_{1i} M_i)S + X) & = & \text{Expand}(Z) \\ \beta_2 & - & \beta_1 & = & \alpha \end{cases}$$

Si le système n'a pas de solution (β_1, β_2), on recommence un petit nombre de fois avec une autre graine $\overline{STXZ}$. Ensuite dans chaque itération :

1. Le Prouveur envoie au Vérifieur :

$$\xrightarrow{\hspace{4cm}}$$
$$H(\overline{STXZ}),\ H(TN_1S + X),\ H(TN_2S + X - TM_0S)$$

2. Le Vérifieur choisi une question $\mathcal{Q}$, en prenant $\mathcal{Q} = 0$ avec probabilité $1/2$, ou $\mathcal{Q} \in \{1, 2\}$, chacune avec probabilité $1/4$. Il envoie $\mathcal{Q}$ à P.

$$\xleftarrow{\hspace{4cm}}$$
$$\mathcal{Q} \in \{0, 1, 2\}$$

3. Si $\mathcal{Q} = 0$, le Prouveur va révéler les valeurs suivantes :

$$\xrightarrow{\hspace{4cm}}$$
$$TMS,\ Z$$

Vérification $\mathcal{Q} = \mathbf{0}$: Le Vérifieur va calculer les $(TN_1S + X)$ et $(TN_2S+X-TM_0S)$. Ensuite il va accepter si $H(TN_1S+X)$ et $H(TN_2S+X - TM_0S)$ sont corrects, et si $Rank(TMS) = r$.

3' Dans les cas $\mathcal{Q} = 1, 2$, le Prouveur va révéler :

$$\xrightarrow{\hspace{4cm}}$$
$$\overline{STXZ},\ \beta_{\mathcal{Q}}$$

Vérification $\mathcal{Q} = \mathbf{1, 2}$: Le Vérifieur vérifie si S et T sont inversibles et si $H(\overline{STXZ})$ est correct. Ensuite il calcule

$$TN_{\mathcal{Q}}S = \sum \beta_{\mathcal{Q}i}\ \ TM_iS$$

et vérifie la valeur de $H(TN_1S + X)$ ou $H(TN_2S + X - TM_0S)$.

Chapitre 26

Le schéma MinRank en pratique

26.1 Performances du schéma

26.1.1 La complexité en communication

On va supposer que les valeurs de hachage sont calculées avec SHA-1. Ceci demande $3 \cdot 160 + 2$ bits pour les deux premières étapes (passes).
On remarque que les valeurs de $\overline{STX} = (S, T, X)$ n'ont pas besoin d'être transmises, elle peuvent être générées avec un générateur pseudo-aléatoire avec une graine de 160 bits, en utilisant la méthode décrite dans 25.1.1.
La dernière étape (passe) demande $2n\eta \log_2 q$ bits dans le cas $\mathcal{Q} = 0$. Dans les deux autres cas elle demande $160 + m \log_2 q$ bits. La moyenne pondérée des complexités en bits pour l'ensemble du schéma est donc :

$$3 \cdot 160 + 2 + \frac{2}{3} \cdot 160 + \frac{2}{3}(n\eta + m) \log_2 q.$$

Ceci doit être multiplié par le nombre d'itérations du schéma qui est 35 pour atteindre la probabilité globale de fraude de 10^{-6}, sachant que la probabilité de fraude de 2/3 dans chacune des itérations.

Version MinRank v2

Dans le chapitre 25.5 on montre comment obtenir 1/2 au lieu de 2/3. On va directement calculer les communications de la version améliorée de 25.7. Elle est de $2 \cdot 160 + 2 + n\eta \log_2 q + 160$ bits pour $\mathcal{Q} = 0$ et $2 \cdot 160 + 2 + 160 + m \log_2 q$ bits sinon. La moyenne pondérée vaut :

$$\left(3 \cdot 160 + 2 + \frac{n\eta + m}{2} \log_2 q\right) \cdot \#\text{itérations}$$

Améliorations supplémentaires

Une analyse plus approfondie indique que dans la pratique 160 bits peut s'avérer trop ou trop peu. Supposons que la sécurité vis à vis des attaques sur MinRank du schéma est 2^{SF}. On va utiliser la longueur des hachés et des graines aléatoires de

$$\frac{3}{2} SF \ \text{ bits.}$$

En effet, le développement de l'informatique semble indiquer que pour le prix d'une puissance de calcul de 2^{SF} on arrive à acquérir $2^{SF/2}$ de mémoire. Un attaquant qui dispose d'une puissance de calcul de 2^{SF} et de capacité de faire des tables de précalculs de taille $2^{SF/2}$, peut s'arranger pour inverser une parmi 2^{SF} valeurs d'une mise en gage avec une fonction de hachage. La même analyse s'applique pour obtenir une sortie d'un générateur pseudo aléatoire souhaitée. Il est facile à voir que remplacer 160 bits par $SF + SF/2$ bits est précisément la bonne solution pour ce niveau de sécurité. Ainsi on arrive à la complexité en communication de

$$\left(\frac{9}{2} SF + 2 + \frac{n\eta + m}{2} \log_2 q\right) \cdot \#\text{itérations}$$

Ce n'est pas tout. Il est possible de chaîner les graines aléatoires du schéma (mais pas les hachés) de sorte à diminuer encore les communications. On peut se contenter d'une seule graine aléatoire A_0 de $\frac{3}{2}SF$ bits pour l'ensemble de schéma. A chaque graine successive A_i du schéma actuel on ajoute va calculer

$$A_i = H(A_0 || i || b_1, \ldots, b_7)$$

avec 7 bits aléatoires b_i car les graines utilisées dans le schéma marchent avec probabilité différente de 1 et parfois il sera nécessaire de choisir une autre graine en essayant avec un autre choix des b_i. Ainsi on va avoir $2^7 = 128$ essais pour avoir une probabilité négligeable de ne jamais arriver à trouver une bonne graine. Il faut aussi que la divulgation de la graine principale de $\frac{3}{2}SF$ bits ne se fasse que à la fin. Ce n'est qu'après toutes les vérifications pour toutes les passes du schéma sont faites. Ainsi, mis à part A_0, chaque étape nécessite seulement $3SF + 7 + 2 + \frac{n\eta+m}{2} \log_2 q$ bits. Ainsi on arrive à la complexité en communication de

$$\frac{3}{2} SF + \left(3SF + 9 + \frac{n\eta + m}{2} \log_2 q\right) \cdot \#\text{itérations}$$

26.1.2 Comparaison avec d'autres schémas

Les schémas à divulgation nulle de connaissance (Zéro-knowledge) à base des problèmes NP-complets sont aujourd'hui très pratiques, mais moins performants que des schémas basés sur la factorisation ou le logarithme discret tels que Fiat-Shamir [157], Schnorr [160] ou Guillou-Quisquater [158, 159]. Par contre leur sécurité théorique est d'un nettement ordre supérieur. Pour cette raison il convient de comparer ces schémas entre eux.

	PKP Shamir	SD Stern	Chen [131] Chen	CLE Stern	PPP Pointcheval	MinRank (A) Author
matrix	16 x 34	256 x 512	32 x 16	24 x 48	101 x 117	6 x 6
field	$\mathbb{F}_{251}$	$\mathbb{F}_2$	$\mathbb{F}_{65535}$	$\mathbb{F}_{257}$	$\mathbb{F}_2$	$\mathbb{F}_{65521}$
passes	5	3	5	3/5	3/5	3
impersonation probability	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{2}{3}/\frac{1}{2}$	$\frac{3}{4}/\frac{2}{3}$	$\frac{2}{3}/\frac{1}{2}$
rounds	20	35	20	35/20	48/35	35/20
impersonation global	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
public key [bits]	272	256	256	80	149	735
secret key [bits]	128	512	512	80	117	160
best attack	2^{60}	2^{70}	2^{53}	2^{73}	2^{61}	$\mathbf{2^{106}}$
bits send/round	665	954	1553	940/824	896/1040	1075/694
global [Kbytes]	**1.62**	4.08	3.79	4.01/**2.01**	5.25/4.44	4.6/**1.94**

Il s'avère que MinRank est un de plus performants parmi les schémas connus.

26.1.3 La rapidité

Les multiplications matricielles du schéma sont des opérations extrêmement rapides à effectuer et peuvent être implementés dans la carte à puce la moins chère.

Septième partie

Les attaques sur les langues naturelles.

Chapitre 27

Attaques avec langues naturelles

Dans cette partie, majoritairement rédigée en anglais, on présente une méthode d'attaque sur les cryptosystèmes multivariables basée sur le fait que le clair est dans une langue naturelle (français, anglais) et codé sur octets.

L'exemple analysé est la cryptanalyse de schéma de Harari mais on peut l'utiliser pour attaquer n'importe quel schéma multivariable. En fait on n'utilise rien du schéma de partage de secret de Harari à part le fait que le calcul de parts se fait par des fonctions multivariables, en l'occurrence linéaires car elles viennent d'un code correcteur linéaire.

La cryptanalyse consiste à combiner les équations supplémentaires qui existent pour les langues naturelles avec les équations du cryptosystème.

On a trouvé qu'il existe des équations quadratiques intéressantes sur des langues naturelles, ce qui a priori n'a rien d'évident. Nous avons explicitement calculé de telles équations pour le français et l'anglais.

27.1 Introduction

27.1.1 Breaking the Harari secret sharing scheme

The Harari secret sharing scheme has been proposed in [73]. For a random secret it's security is unconditional. The chief advantage of the Harari scheme is the possibility to chose a required security threshold and keep the expansion rate small, unlike the Shamir perfect sharing scheme. With 64 or more missing share bits it is simply secure, there is not enough information neither to determine nor to recover the secret.

In this section we study the security of the scheme when the secret is not random. We show relations between the file compression rate and security. We show how to chose a parameters to achieve provable security for a given compression rate. Unfortunately it gives big parameters that in the case of very redundant files tend towards the very expansive perfect schemes.

However a simple compression of the file restores all the advantages of Harari scheme.
We show that attacking incompressible files is difficult, as it would lead to unexpected progress in file compression.

We propose several realistic attacks of the scheme without pre-compression. A text file can be recovered with up to 320 missing bits.
We show that it is always possible to recover a secret file provided a few fragments of it, of total size close to the missing information. It works also for random or compressed files.
If a secret file fragment has been independently re-shared, and we know some shares, it is still possible to break the scheme. It works also if some shares are re-shared in a tree-like structure. Finally, we show that for some redundant files, as with a lot of parity bits, blank spaces, periodic patterns or repetition of large blocks, the secret can be recovered without any additional information.

27.1.2 Beyond the secret sharing

The main interest of this part is not the cryptanalysis of secret sharing, but that the equations we obtained here for English or French, can be used to attack any multivariate cryptosystem if we suspect that the cleartext is in plain text or some other kind of redundant file.

27.2 Redundancy and compression

Let S be a secret file. We can formalise any kind of 'a priori' knowledge on S saying that $S \in_{\mathcal{L}} L$, L being a language on an alphabet $\mathbf{K}$, with an associated probability distribution $\mathcal{L}$.

Let $\mathcal{L}^n$ be the probability distribution of all possible n-symbols substrings of S. Let H(X) denote the Shannon's entropy of a random variable X.

The entropy of a language $\mathcal{L}$ is

$$H_{\mathcal{L}} \stackrel{def}{=} \lim_{n\to\infty} \frac{H(\mathcal{L}^n)}{n}.$$

It gives the best possible compression rate for $S \in \mathcal{L}$:

$$C_S^{min} \stackrel{def}{=} H_{\mathcal{L}}/log_2 q$$

with $q = \#L^1 = \#K$ being the alphabet cardinal.
The compression rate is related to classical redundancy ρ :

$$\rho = 1 - C.$$

The classical entropy theory shows that

$$C_S \geq C_S^{min} = H_{\mathcal{L}}/log_2 q,$$

and that the limit can be achieved as close as we want with Huffman coding.

For example if S is a text file in English and if characters are coded on 8 bits, we have the following estimation following Stinson [5] :

$$C_{text}^{min} = \frac{H_{English}}{8} \approx 0.16$$

and at any rate, the lower bound is estimated in [5] to be,

$$C_{text}^{min} = \frac{H_{English}}{8} \geq 0.125$$

If we compress a text by standard methods as .zip or .arj file (Huffman compression), we get :

$$C_{text} \approx 0.33 - 0.38.$$

The difference is due to the fact that current compression methods are required to be fast and to work for all files, not only for text files in English.

It is obvious that the ideal compression rate of about 0.16 can be achieved by Huffman compression, just dealing with blocs large enough and provided with enough context information of English. It requires however extremely slow and complex algorithms, so let's assume a realistic value

$$C_{text} = 0.20.$$

27.3 Application to secret sharing

We assume that the 'a priori' knowledge of the secret in Harari scheme gives a compression algorithm with the compression rate C_S.

Let K be a secret file we shared with a linear secret sharing scheme, such as Harari scheme [73].

Assuming the secret length K is big, the entropy of the secret is

$$H(S) = C_S K \log_2 q.$$

With missing r among K symbols necessary to recover S, there are about

$$\frac{2^{H(S)}}{2^{(K-r)}} = 2^{(r-K(1-C_S))\log_2 q}$$

possible solutions S. It means that :

Proposition 27.3.0.1 *The Harari scheme is secure if*

$$(r_{min} - K(1 - C_S^{min}))\log_2 q > 64, \qquad r_{min} \leq K$$

i.e.

$$K \geq r_{min} \geq K(1 - C_S^{min}) + \frac{64}{\log_2 q}(1)$$

Example 27.3.0.2 *For a text file in English without any additional information we have seen that $C_S^{min} \geq 0.125$. Therefore for*

$$r > r_{min}^{English} = 0.875K + \frac{64}{\log_2 q}$$

it is secure.

Corollary 27.3.0.3 *If K is big and $C_S^{min} \to 0$ then r_{min} tends to the perfect scheme part size K.*

This is simply a version of Shannon theorem about perfect secrecy. Each part (length r) of a perfect secret sharing scheme guarantee a perfect safety of the secret (length K), therefore $r \geq K$.

Such r values are unfortunately too big to be used in practice.

27.3.1 Using compressed files

Is Harari scheme as bad as the perfect schemes for very redundant files ? Not at all. We can simply compress the file before sharing it.

For a random file ($C_S^{min} = 1$) it's enough to choose

$$r \geq r_{min}^{rand} = \frac{64}{\log_2 q}.(2)$$

The question is if it is still secure if a file is not random but just looks like, which is the case of a compressed file with C_S close to 1.

Proposition 27.3.1.1 *If an effective algorithm exist to recover an incompressible file S with some noticeable probability, provided the insufficient $K-r$ share symbols, it would give a new compression algorithm which can compress a file already compressed by traditional methods.*

The compression algorithm would be simply the $K - r$ Reed-Solomon code equations of the sharing process or identity, depending if the cryptanalysis works in the given case or not.

If the system can be broken, it gives a small correction to the compression rate of a file. The difference $C_S^{min} - C_S$ cannot be important, because as we have seen in the previous section, the lower limit to the compression rate for a given probability distribution $\mathcal{L}$ is fixed and equal $C_S^{min} = H_{\mathcal{L}}/log_2 q$.

The problem is to compute the actual value of C_S^{min}. It is usually done by one of two methods :

The first method is to find a better compression algorithm and it is very difficult because it is effective.

The second is to formalise as restrictively as possible what the secret can be. It is still very difficult, for example if the text is in English, such a formalisation requires not less than an extensive knowledge of english syntax, grammar, civilization etc. And still we need to compute the entropy of such a very complex distribution.

We remark that both methods give only un upper bound on the C_S^{min} value, and it will give an upper bound on the attacking complexity of Harari scheme. This situation is frequent in cryptography but here, we know that a fixed, limit complexity exists.

As the scheme is secure for C_S close to 1, most of our attacks assume that $C_S < 1/2$.

27.4 Attacks

Let $S = (s_1, s_2, \ldots, s_K), s_i \in \mathbf{F}_q$ be a secret file. The shares in the Harari scheme are computed as linear equations of K variables over $\mathbf{F}_q$.

$$X = \begin{cases} x_1 = \sum_k^{i=1} \gamma_i^1 s_i \\ \vdots \\ x_r = \sum_k^{i=1} \gamma_i^r s_i \end{cases}$$

These equations come from the generating matrix G of a Reed-Solomon code. We do not use their structure a lot. We use mainly the fact that they are linear, and occasionally that they are cyclic and have a fixed minimal distance. Further algebraic properties could improve our attacks and even lead to new ones, especially if shares are chosen as contiguous blocks of a codeword. Our attacks work all the same whatever the position and choice of shares is.

The security specification of the scheme states that with one missing share it is difficult to recover a secret (there is not enough information). A share has r symbols in $\mathbf{F}_q$ and it's size is $r \log_2 q$ bits.

Therefore a standard attack assumption would be that one share is missing. Since the shares are simply a fragments of a codeword generated from the secret, our attacks will work exactly the same way if more shares are missing, or some symbols of different shares are unknown. For all attacks, we call r the total number of missing $\mathbf{F}_q$ symbols from all shares of the secret file S. Therefore there are $K - r$ known share symbols.

To recover S we need simply to solve $K - r$ linear equations with K variables over $K = \mathbf{F}_q$ (the coefficients γ_l^k of these equations are public).

Our first attacks focus on finding additional r equations so that the secret can be recovered by gaussian reduction.

Obvious attacks - some variables can be guessed.

If a file has unused bits that are fixed to 0 or 1, e.g. a text file using only 7 bit characters, and if they are at least r such bits, then we have $K - r$ equations with less than $K - r$ variables. In the example with 7-bit text file, if $r \leq K/8$ the scheme is broken.

Let us suppose we share a data file, program or image in which we expect a large block filled with 0's or spaces, or just any known block. It is easily broken provided the length of the block is at least r. The place in which the block occurs is found by exhaustive search, and it is still practicable for 2 or 3 blocks of total length r.

Finally, a limited, fixed number of variables can be guessed in every file by exhaustive search. This number is about $\frac{40}{C_S \log_q}$ provided a really working compression algorithm with the rate C_S.

Let B (Bonus) be the number of variables hat can be guessed.

Therefore our attacks give us the following bound on the r_{min} :

$$r_{min} > \frac{64/C_S + B}{\log_2 q}$$

Are they better attacks that improve this bound ?

27.4.1 Attacks with linear relations between variables.

The same attack and bound applies when B is the number of linear equations that we guessed on the x_i.

Our simulations show many equations true with probability close to 1 on the bits of a character taken out of a plain text.

The most effective way to use such relations, is to suppose they are true for some random characters in the text.

Example :

For plain english we have $b_7 + b_8$ with probability $p = 0.766$. We may guess 50 places such that it happens with probability $p^{50} = 2^{-19}$ instead of 2^{-50}.

27.4.2 Attacks with quadratic relations between variables.

We can also use the quadratic equations, and use XL to solve the set of equations we get.

We discovered that there indeed such non-trivial quadratic equations on plain text, which was not expected.

27.5 Simulations for English

The tables show equations on bits $b_1, \ldots, b_8$ of a character. We show those that are true with a probability p that differs a lot form 1/2, and also the entropy H_p of such an equation.

We do not give all the equations that can be found, but only a basis for constructing linearly independent sets with a minimum entropy.

Equation	p	H_p
$b_8 = 0$	0.9999687	0.00051
$b_6 = 1$	0.9603613	0.24063
$b_2 + b_3 + b_4 + b_7 = 0$	0.7950279	0.73176
$b_7 = 1$	0.7772594	0.76514
$b_5 = 0$	$0,7573305$	0.79945
$b_4 + b_5 + b_6 + b_7 = 1$	0.6856772	0.89810
$b_1 + b_2 + b_4 + b_5 + b_7 = 0$	0.7018489	0.87902

Equation	p	H_p
$b_5 + b_5b_7$	0.0013776	0.01508
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9969316	0.03004
$b_5 + b_5b_6$	0.0058707	0.05196
$b_2 + b_2b_6 + b_3 + b_3b_6 + b_4 + b_4b_6 + b_6b_7 + b_7$	0.0064969	0.05655
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_7$	0.0080780	0.06776
$b_5b_6 + b_5b_7$	0.0072483	0.06194
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_4 + b_4b_7$	0.0074518	0.06338
$b_2 + b_2b_6 + b_3 + b_3b_7 + b_4b_6 + b_4b_7 + b_6$	0.9897459	0.08247
$b_1 + b_1b_6 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_6 + b_6$	0.9893545	0.08504
$b_2 + b_2b_6 + b_3 + b_3b_7 + b_4b_6 + b_4b_7 + b_5 + b_5b_7 + b_6$	0.9903721	0.07832
$b_1 + b_1b_6 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5b_6 + b_5b_7 + b_6$	0.9899807	0.08092
$b_2 + b_2b_6 + b_3 + b_3b_6 + b_4 + b_4b_6 + b_5 + b_5b_7 + b_6b_7 + b_7$	0.0078745	0.06635
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_6 + b_6b_7 + b_7$	0.9905756	0.07695
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_5 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9897616	0.08237
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_6 + b_6 + b_6b_7 + b_7$	0.9915619	0.07025
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5b_6 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9910609	0.07367
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_4 + b_4b_7$	0.0191149	0.13644
b_4b_5	0.0179407	0.12972
$b_3 + b_3b_7 + b_4 + b_4b_7$	0.0146219	0.11007
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_6$	0.0133225	0.10209
$b_4b_5 + b_5 + b_5b_7$	0.0172832	0.12590
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_4 + b_4b_7 + b_5 + b_5b_7$	0.0182695	0.13161
$b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_7$	0.0139956	0.10625
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_5b_6 + b_5b_7$	0.0139487	0.10596
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_6 + b_6b_7 + b_7$	0.9799771	0.14157
$b_3 + b_3b_7 + b_4 + b_4b_7 + b_6 + b_6b_7 + b_7$	0.9854251	0.10979
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_7 + b_6 + b_6b_7 + b_7$	0.9974326	0.02579
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_5 + b_4b_7 + b_6 + b_6b_7 + b_7$	0.9800554	0.14113
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_5 + b_5b_6 + b_6 + b_6b_7 + b_7$	0.9847050	0.11414
$b_2 + b_2b_7 + b_3 + b_3b_7 + b_5 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9803842	0.13928
$b_3 + b_3b_7 + b_4 + b_4b_7 + b_5 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9860513	0.10596
$b_1 + b_1b_7 + b_3 + b_3b_7 + b_4 + b_4b_5 + b_4b_7 + b_5 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9804625	0.13884
$b_1 + b_1b_7 + b_2 + b_2b_7 + b_5b_6 + b_5b_7 + b_6 + b_6b_7 + b_7$	0.9838909	0.11900

Source text was : The Canterville Ghost, by Oscar Wilde.

27.6 Simulations for French

They depend a lot on the character encoding and the origin of the file.

Equation	p	H_p
$b_6 + b_7 = 0$	1.0000000	1.00000
$b_8 = 1$	0.0175783	0.12762
$b_6 = 1$	0.8980457	0.47516
$b_7 = 1$	0.8980457	0.47516

Bibliographie

Généralités, ouvrages et articles de référence

Cryptographie et Cryptologie

[1] Gilles Brassard : "Cryptologie Contemporaine" ; Masson 1993.

[2] James Ellis : "The story of non-secret encryption" ; Available at `http://www.cesg.gov.uk/about/nsecret/`

[3] Alfred J. Menezes, Paul C. van Oorshot, Scott A. Vanstone : *Handbook of Applied Cryptography* ; CRC Press.

[4] Bruce Schneier : *Applied Cryptography* ; Wiley and sons.

[5] Douglas R. Stinson : *Cryptography, theory and practice* ; CRC Press 1995.

[6] Jacques Stern : *La science du secret* ; Éditions Odile Jacob, Paris, 1998, www.odilejacob.fr.

[7] Jacques Patarin : *La Cryptographie Multivariable* ; Mémoire d'habilitation à diriger des recherches de l'Université Paris 7, 1999.

[8] Jacques Patarin, Louis Goubin, Nicolas Courtois, + papers of Eli Biham, Aviad Kipnis, T. T. Moh, et al. : *Asymmetric Cryptography with Multivariate Polynomials over a Small Finite Field* ; connu en tant que 'poly orange', la compilation des versions mises à jour de différents articles de cryptographie multivariable, avec des remarques et une liste de schémas connus. Disponible auprès de J.Patarin@frlv.bull.fr.

[9] Claude Elmwood Shannon, Collected Papers, Sloane & Wyner eds., New York, IEEE Press, 1993.

La théorie de la complexité

[10] Gilles Brassard : "A note on the complexity" ; IEEE Tran. Inform. Theory, Vol. IT-25, 1979, pp. 232-233.

[11] Michael Garey, David Johnson : *Computers and Intractability, a guide to the theory of NP-completeness*, Freeman, p. 251.

Conversions et sécurité prouvable des cryptosystèmes à clef publique

[12] Mihir Bellare, Anand Desai, David Pointcheval, and Philip Rogaway : *Relations among Notions of Security for Public-Key Encryption Schemes ;* In Crypto'98, LNCS 1462, pages 26-45. Springer-Verlag, Berlin, 1998.

[13] M. Bellare and P. Rogaway : *Optimal Asymmetric Encryption How to Encrypt with RSA ;* In Advances in Cryptology, Eurocrypt 94, pp. 92-111. Springer Verlag, 1994.

[14] R. Canetti, O. Goldreich and S. Halevi : *The random oracle methodology, revisited (preliminary version) ;* ACM symposium on Theory of computing, ACM, pp. 209–218, May 24 - 26, 1998, Dallas, TX USA.

[15] E. Fujisaki and T. Okamoto : *How to enhance the security of public-key encryption at minimum cost ;* IEICE Transaction of Fundamentals of electronic Communications and Computer Science, E83-A(1) :24–32, January 2000.

[16] E. Fujisaki and T. Okamoto : *Secure Integration of Asymmetric and Symmetric Encryption Schemes ;* In Crypto'99, LNCS 1666, pages 537-554. Springer-Verlag, Berlin, 1999.

[17] T. Okamoto and D. Pointcheval : *REACT : Rapid Enhanced-security Asymmetric Cryptosystem Transform ;* In the Cryptographers' Track of the RSA Security Conference '2001, LNCS2020, Springer-Verlag, Berlin, 2001.

[18] Victor Shoup : *OAEP Reconsidered ;*Preprint, November 2000. Available from `http://eprint.iacr.org/`.

Les signatures numériques

[19] S. Goldwasser, S. Micali and R. Rivest : *A Secure Digital Signature Scheme ;* Siam Journal on Computing, Vol. 17, 2 (1988), pp. 281-308.

[20] Ralph C. Merkle : *A digital signature based on a conventional encryption function ;* Crypto'87, LNCS 293, pp.369-378.

[21] D. Pointcheval, J. Stern : *Security arguments for Digital signatures and Blind Signatures ;* Journal of Cryptology, Vol.13(3), Summer 2000, pp. 361-396.

[22] Ronald R. Rivest, Adi Shamir and Yael Tauman : *How to leak a secret* ; Asiacrypt 2001, LNCS, Springer-Verlag.

Des fonctions à sens unique

[23] Michael Luby, *Pseudorandomness and Cryptographic Applications*; Princeton University Press, 1996.

[24] R. Canetti : *Towards Realizing Random Oracles : Hash Functions that Hide All Partial Information*, R. In Advances in Cryptology, Crypto'97, Springer-Verlag.

[25] See also [14].

Mathématiques Appliquées

[26] I. Blake, X. Gao, R. Mullin, S. Vanstone and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publishers.

[27] H. Cohen : *A Course in Computational Algebraic Theory*; Springer Verlag 1993.

[28] F. R. Gantmacher . : *The theory of matrices*; Chelsea Publishing Company, Volume one, chapter VII.

[29] Rudolf Lidl, Harald Niederreiter : *Finite Fields*; Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press.

[30] G. Mullen, *Permutation Polynomials over Finite Fields*; in *Finite Fields, Coding Theory, and Advances in Communications and Computing*; Dekker, Volume 141, 1993, pp. 131-152.

Courbes Elliptiques en cryptographie

[31] Jerzy Gawinecki, Janusz Szmidt : *Zastosowanie ciał skończonych i krzywych eliptycznych w kryptografii ;* Wojskowa Akademia Techniczna, Bel Studio, Warszawa.

[32] V.S. Miller : *Use of elliptic curves in cryptography*; Crypto'85, LNCS 218.

[33] Neal Koblitz : *Elliptic curve cryptosystems*; Mathematics of Computation, 48 (1987), pp. 203-209.

[34] Michael Rosing : *Elliptic Curve Cryptography*; Manning Publications, USA. The book can be ordered at it's web site, which contains also it's preface, index, the chapter 5, etc. with all the C-programs from the book.

[35] Mike Wiener, Robert Zuccherato : *Faster Attacks on Elliptic Curve Cryptosystems*; SAC'98, Ontario, Canada.

Résolution d'équations

Équations univariables sur des corps finis

[36] Alfred J. Menezes, Paul C. van Oorshot, Scott A. Vanstone : *Some computational aspects of root finding in* $GF(q^m)$; in Symbolic and Algebraic Computation, Lecture Notes in Computer Science, 358 (1989), pp. 259-270.

[37] Paul van Oorshot and Scott Vanstone : *A geometric approach to root finding in* $GF(q^m)$; IEEE Trans. Info. Th., 35 (1989), pp. 444-453.

[38] J. von zur Gathen and Victor Shoup, *Computing Fröbenius maps and factoring polynomials* ; Proceedings of the 24th Annual ACM Symposium in Theory of Computation, ACM Press, 1992.

[39] La librairie NTL pour la théorie des nombres et factorisation des polynômes. Elle est écrite en C++, portable, et gratuite pour un usage non-commercial. Disponible sur `www.shoup.net`.

Équations multivariables sur des corps finis

[40] Iyad A. Ajwa, Zhuojun Liu, and Paul S. Wang : *Gröbner Bases Algorithm* ; ICM Technical Reports, February 1995, see `http://symbolicnet.mcs.kent.edu/icm/reports/index1995.html`.

[41] Don Coppersmith : *Finding a small root of a univariate modular equation* ; Proceedings of Eurocrypt'96, Springer-Verlag, pp.155-165.

[42] Nicolas Courtois, Adi Shamir, Jacques Patarin, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, In Advances in Cryptology, Eurocrypt'2000, LNCS 1807, Springer-Verlag, pp. 392-407.

[43] Jean-Charles Faugère : *A new efficient algorithm for computing Gröbner bases* (F_4).Journal of Pure and Applied Algebra 139 (1999) pp. 61-88. See www.elsevier.com/locate/jpaa

[44] Jean-Charles Faugère : *Computing Gröbner basis without reduction to 0* ; technical report LIP6, in preparation, source : private communication.

[45] Willi Meier, Jean-Daniel Tacier et al. : *On the security of Multivariate Signature scheme* ; work in progress on a new method of solving multivariate underdefined systems of equations in finite fields, title might change.

[46] Peter L. Montgomery : A Block Lanczos Algorithm for Finding Dependencies over GF(2) ; Eurocrypt'95, LNCS, Springer-Verlag.

[47] Bart Selman, Hector Levesque, David Mitchell : *A New Method for Solving Hard Satisfiability Problems* ; In Proceedings 11th National Conference on Artificial Inteligence, 1992.

La cryptographie multivariable quadratique

[48] H. Fell, W. Diffie, *Analysis of a public key approach based on polynomial substitutions*, in Advances in Cryptology, Proceedings of Crypto'85, LNCS n° 218, Springer-Verlag, 1985, pp. 340-349.

Le cryptosystème de Matsumoto et Imai

[49] E. Brickell, A. Odlyzko, *Cryptanalysis, A Survey of Recent Results*, p. 506, in *Contemporary Cryptology*, IEEE Press, 1992, edited by Gustavus J. Simmons.

[50] Hans Dobbertin, *Almost Perfect Nonlinear Power Functions on* $GF(2^n)$; paper available from the author.

[51] T. Matsumoto, H. Imai, H. Harashima, H. Miyakawa : *Asymmetric cryptosystems using obscure representations of enciphering functions* ; 1983 Natl. Conf. Rec. On Inf. Syst., IECE Japan, S8-5, September 1983 (in Japanese).

[52] Tsutomu Matsumoto, Hideki Imai :*A class of asymmetric cryptosystems based on polynomials over finite rings* ; 1983 IEEE International Symposium on Information Theory, Abstract of Papers, pp.131-132, September 1983.

[53] Tsutomu Matsumoto, Hideki Imai :*Algebraic Methods for Construction of Asymmetric Cryptosystems* ; AAECC-3, Grenoble, 15-19 juin 1985.

[54] Tsutomu Matsumoto, Hideki Imai : *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption* ; EUROCRYPT'88, Springer-Verlag 1998, pp. 419-453.

[55] Isabelle Guerin-Lassous : *Rapport de Stage-Étude et attaque de l'algorithme de Matsumoto-Imai et de ses généralisations* ; DEA Intelligence Artificielle et Algorithmique, Université de Caen.

[56] Jacques Patarin : *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88* ; Crypto'95, Springer-Verlag, pp. 248-261.

Généralisations et applications de Matsumoto et Imai

[57] Neal Koblitz : *Algebraic aspects of cryptography* ; Springer-Verlag, ACM3, 1998, Chapter 4 *Hidden Monomial Cryptosystems*, pp. 80-102.

[58] Nicolas Courtois : *Cryptographie Asymétrique et l'isomorphisme des polynômes (IP)* ; Rapport de Magistère MMFAI, Université Paris 6, École Normale Supérieure, Novembre 1997.

[59] Jacques Patarin, Nicolas Courtois, Louis Goubin : *C*-+ and HM - Variations around two schemes of T. Matsumoto and H. Imai* ; Asiacrypt'98, Springer-Verlag.

[60] Jacques Patarin : *Asymmetric Cryptography with a Hidden Monomial* ; Crypto'96, Springer Verlag, pp. 45-60.

[61] Jacques Patarin, Louis Goubin, Nicolas Courtois : *Flash, a fast multivariate signature algorithm*; Cryptographers' Track RSA Conference 2001, San Francisco 8-12 Avril 2001, LNCS2020, Springer-Verlag. Also published in Proceedings of the First Open NESSIE Workshop, 13-14 November 2000, Leuven, Belgium.

[62] Flash and Sflash, submitted to Nessie European call for cryptographic primitives `http://www.cryptonessie.org`. The official web page of Flash is `http://www.minrank.org/flash/`. The official web page of Sflash is `http://www.minrank.org/sflash/`.

[63] R. Steinwandt, W. Geiselmann, and Th. Beth *Attacking the Affine Parts of SFLASH*; to appear in Proceedings of Eighth Cryptography and Coding, Lecture Notes in Computer Science, Springer. Also presented at the 2nd NESSIE workshop, LNCS, Springer.

[64] R. Steinwandt, W. Geiselmann, and Th. Beth *A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates FLASH and SFLASH*; to appear in Proceedings of ISC '01, Lecture Notes in Computer Science, Springer. Also presented at the 2nd NESSIE workshop, LNCS, Springer.

Le problème HFE et applications

[65] Jacques Patarin : *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP) : two new families of Asymmetric Algorithms*; Eurocrypt'96, Springer Verlag, pp. 33-48.

[66] Jacques Patarin, Louis Goubin, Nicolas Courtois : Quartz, **1**28-*bit long digital signatures*; Cryptographers' Track RSA Conference 2001, San Francisco 8-12 Avril 2001, LNCS2020, Springer-Verlag. Also published in Proceedings of the First Open NESSIE Workshop, 13-14 November 2000, Leuven, Belgium.

[67] Quartz, submitted to Nessie European call for cryptographic primitives `http://www.cryptonessie.org`. The official web page of Quartz is `http://www.minrank.org/quartz/`.

[68] Nicolas Courtois : *Generic attacks and provable security of Quartz*; work in progress, presented at the Nessie workshop, September 13th 2001, Royal Holloway, University of London.

[69] Nicolas Courtois : *The security of Hidden Field Equations (HFE)*; Cryptographers' Track RSA Conference 2001, San Francisco 8-12 Avril 2001, LNCS2020, Springer-Verlag.

[70] Nicolas Courtois : *On multivariate signature-only public key cryptosystems;* To appear in 2001.

[71] Nicolas Courtois : HFE security, the HFE cryptosystem web page. `http://hfe.minrank.org`

[72] Adi Shamir, Aviad Kipnis : *Cryptanalysis of the HFE Public Key Cryptosystem*; In Advances in Cryptology, Proceedings of Crypto'99, Springer-Verlag, LNCS. It can be found at `http://www.minrank.org/~courtois/hfesubreg.ps`

D'autres cryptosystèmes multivariables

[73] Louis Goubin, Jacques Patarin : *Trapdoor one-way permutations and multivariate polynomials* ; juin 1997, à paraître dans ISICS'97, Springer-Verlag.

[74] Sami Harari : "Secret sharing with MDS codes", available from the author.

[75] Aviad Kipnis, Adi Shamir : *Cryptanalysis of Oil and Vinegar Signature Scheme* ; Crypto'98, Springer-Verlag.

[76] Adi Shamir : *Efficient signature schemes based on birational permutations* ; Crypto'93, Springer-Verlag, pp1-12.

[77] Aviad Kipnis, Jacques.Patarin, Louis Goubin : Louis Goubin, Kipnis Aviad, Jacques Patarin : *Unbalanced Oil and Vinegar Signature Schemes* ; Eurocrypt 1999, Springer-Verlag.

[78] Don Coppersmith, Jacques Stern, Serge Vaudenay : *Attacks on the birational permutation signature schemes* ; Crypto'93, Springer-Verlag, pp. 435-443.

[79] Don Coppersmith, Jacques Stern, Serge Vaudenay, *The Security of the Birational Permutation Signature Schemes*, in Journal of Cryptology, 10(3), pp. 207-221, 1997.

Les cryptosystèmes triangulaires T, TPM, TTM

[80] The (unofficial) web page for TTM. `http://www.minrank.org/ttm/`

[81] Louis Goubin, Nicolas Courtois *Cryptanalysis of the TTM Cryptosystem* ; Advances of Cryptology, Asiacrypt'2000, 3-9 December 2000, Kyoto, Japan, Springer-Verlag.

[82] Louis Goubin, Jacques Patarin : *Asymmetric cryptography with S-boxes* ; June 1997, ISICS'97, Springer-Verlag.

[83] C. Y. Chou, D. J. Guan, J. M. Chen, *A systematic construction of a Q_{2^k}-module in TTM*, Preprint, October 1999.

[84] T.T. Moh, *A public key system with signature and master key functions*, Communications in Algebra, 27(5), pp. 2207-2222, 1999. Available at `http://www.usdsi.com/public.ps`

[85] T.T. Moh, *A fast public key system with signature and master key functions*, in Proceedings of CrypTEC'99, International Workshop on Cryptographic Techniques and E-commerce, Hong-Kong City University Press, pages 63-69, July 1999. Available at `http://www.usdsi.com/cryptec.ps`

[86] *The US Data Security Public-Key Contest*, available at `http://www.usdsi.com/contests.html`

Les problèmes d'isomorphismes

Équivalence des codes

[87] E. Petrank and R. M. Roth. *Is code equivalence easy to decide ?*IEEE Transactions on Information Theory, 43(5), pp. 1602-1604, September 1997.

[88] Nicolas Sendrier : *Finding the Permutation Equivalent Linear Codes : The Support Splitting Algorithm ;* IEEE Trans. on Information Th. July 2000, Vol. 46 Nb. 4, pp. 1193-1203.

Le problème IP

[89] Nicolas Courtois, Louis Goubin, Jacques Patarin : *Improved Algorithms for Isomorphism of Polynomials*; Eurocrypt 1998, Springer-Verlag.

[90] voir aussi [65].

Le problème MP et Tensor Rank

[91] Don Coppersmith, Shmuel Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Computation (1990), **9**, pp. 251-280.

[92] Don Coppersmith, Shmuel Winograd, *On the asymptotic complexity of matrix multiplication*, SIAM Journal Comp., 11(1980), pp 472-492.

[93] John Gustafson, Srinivas Aluru : *Massively Parallel Searching for Better Algorithms or, How to Do a Cross Product with Five Multiplications*; Ames Laboratory, Dept. of Energy, ISU, Ames, Iowa. Accessible à `www.scl.ameslab.gov/Publications/FiveMultiplications/Five.html`.

[94] Johan Håstad, *Tensor Rank is NP-Complete*, Journal of Algorithms, vol. 11, pp. 644-654, 1990.

[95] Julian D. Laderman : "A noncommutative algorithm for multiplying 3x3 matrices using 23 multiplications" ; Bulletin of the American Mathematical Society, Vol. 82, Nr. 1, January 1976.

[96] Victor Pan, *How to multiply matrices faster*; Lectures Notes in Computer Science No179, Edition Springer-Verlag, 1984.

[97] Victor Pan : *Complexity of Computations with Matrices and Polynomials*, j-SIAM-REVIEW, 34, pp 225-262.

[98] Volker Strassen : *Gaussian elimination is not optimal*; Numerishe Mathematik 13, 1969, pp. 354-356.

[99] Volker Strassen, *The asymptotic spectrum of tensors*, J. Reine Angew. Math., vol. 384, pp. 102-152, 1988.

Les problèmes de décodage

La difficulté de décodage du syndrome SD

[100] Alexander Barg : *Handbook of coding theory, Chapter 7 : Complexity Issues in Coding Theory ;* North Holland, 1999.

[101] E.R. Berlekamp, R.J. McEliece, H.C.A. van Tilborg : *On the inherent intractability of certain coding problems*; IEE Trans. Inf. Th., It-24(3), pp. 384-386, May 1978.

[102] Anne Canteaut : *Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes*, thèse de Doctorat, Université Paris 6, soutenue le 10 Octobre 1996.

[103] Anne Canteaut, Florent Chabaud : *A new algorithm for finding minimum-weight words in a linear code : application to McEliece's cryptosystem and to BCH Codes of length 511 ;*

[104] Florent Chabaud : *Asymptotic analysis of probabilistic algorithms for finding short codewords* in Proceedings of Eurocode'92, LNCS, Springer-Verlag.

[105] *Ilya I. Dumer ; Suboptimal decoding of linear codes : partition technique ;* IEEE-IT 42(6), November 1996, pp. 1971 - 1986.

[106] P. J. Lee and E. F. Brickell. *An observation on the security of McEliece's public-key cryptosystem ;* In Advances in Cryptology , Eurocrypt'88, LNCS 330, pp. 275-280. Springer-Verlag, 1988.

[107] Jacques Stern : *A method for finding codewords of small weight* ; Coding Theory and Applications, LNCS 434, pp.173-180, Springer-Verlag.

[108] Voir aussi [115].

[109] Alexander Vardy : *The intractability of computing the minimum distance of a code ; IEEE Transactions on Information Theory, Nov 1997, Vol.43, No. 6 ; pp. 1757-1766.*

[110] D. J. A. Welsh : Combinatorial problems in matroid theory ; in Combinatorial Mathematics and its Applications, D. .J. A. Welsh Editor, London UK, Academic, 1971, pp. 291-307.

Le problème MinRank, et les codes de type rang

[111] Nicolas Courtois and Ernst M. Gabidulin. : *Security of cryptographic schemes based on rank problems*; work in progress.

[112] E. M. Gabidulin. *Theory of codes with maximum rank distance.* Problems of Information Transmission, 21 :1-12, 1985.

[113] Jeffrey O. Shallit, Gudmund S. Frandsen, Jonathan F. Buss : *The Computational Complexity of Some Problems of Linear Algebra problems*, BRICS series report, Århus, Denmark, RS-96-33, available on the net `http://www.brics.dk/RS/96/33/`.

[114] L.G. Valiant : *Completeness classes in algebra.* In Proc. Eleventh Ann. ACM Symp. Theor. Comp., pp. 249-261, 1979.

[115] Jacques Stern, Florent Chabaud : *The cryptographic security of the syndrome decoding problem for rank distance codes.* In Advances in Cryptology, Asiacrypt'96, LNCS 1163, pp. 368-381, Springer-Verlag.

Des functions trappe multivariables linéaires

[116] Anne Canteaut, Nicolas Sendrier : *Cryptanalysis of the Original McEliece Cryptosystem. In Advances in Cryptology ;* Asiacrypt'1998, LNCS 1514, pp.187-199.

[117] Nicolas Courtois, Matthieu Finiasz and Nicolas Sendrier : *How to achieve a McEliece-based Digital Signature Scheme* ; Published as Inria rapport de recherche 4118, February 2001, preprint available at `http://www.minrank.org/mceliece/`

[118] The web page for the McEliece signature scheme. `http://www.minrank.org/mceliece/`

[119] Ernst M. Gabidulin, A. V. Paramonov, O. V. Tretjakov : *Ideals over a Non-Commutative Ring and their Applications in Cryptology. Eurocrypt 1991, pp. 482-489.*

[120] Ernst M. Gabidulin : On public-key cryptosystems based on linear codes, efficiency and weakness ; 4th IMA conference on Cryptography and Coding 1993, IMA press 1995.

[121] Ernst M. Gabidulin, Alexei V. Ourivski : *Improved GPT public key cryptosystems.* Coding, Communications, Broadcasting, Research Studies Press, 2000, pp 73-102.

[122] Ernst M. Gabidulin, Alexei V. Ourivski : *Modified GPT PKC with Right Scrambler.* WCC 2001, Paris, France, Daniel Augot a,nd Claude Carlet Editor.

[123] J. Keith Gibson : *Severely Denting the Gabidulin Version of the McEliece Public Key Cryptosystem.* Designs, Codes and Cryptography 6(1) : 37-45 (1995)

[124] Keith Gibson : *The Security of the Gabidulin Public Key Cryptosystem.* Eurocrypt 1996, LNCS 1070, Springer-Verlag, pp. 212-223

[125] J.K. Gibson : *Equivalent Goppa Codes and Trapdoors to McEliece's Public Key Cryptosystem ;* Eurocrypt'1991, Springer-Verlag, LNCS 547 pp.517-521.

[126] Pierre Loidreau : *Étude et Optimisation des cryptosystèmes à clé publique fondés sur la théorie des codes correcteurs ;*PhD thesis, École Polytechnique, 2001.

[127] Pierre Loidreau : *Some weak keys in McEliece public-key cryptosystem* ; In IEEE International Symposium on Information Theory, ISIT'98, Boston, USA, 1998.

[128] R.J. McEliece : *A public key cryptosystem based on algebraic coding theory* ; DSN Progress Report42-44, Jet Propulsion Laboratory, 1978, pp. 114-116.

[129] Hans Niederreiter : *Knapsack-type cryptosystems and algebraic coding theory* ; In Probl. Contr. and Information Theory, 1986.

[130] Jacques Stern : *Can one design a signature scheme based on error-correcting codes ? ;* Rump session Asiacrypt'1994, LNCS 917, pp.424-426.

Authentification Zéro-knowledge basée sur les problèmes difficiles des codes

[131] Kefei Chen : *Improved Girault Identification scheme*; Electronic Letters, 15 September 1994, Vol. 30, NO... 19, pp. 1590-1591.

[132] Kefei Chen : *A new identification algorithm*. Cryptography Policy and algorithms conference, vol. 1029, LNCS, Springer-Verlag, 1996.

[133] N. Courtois : *The MinRank problem*. MinRank, a new Zero-knowledge scheme based on the NP-complete problem. Presented at the rump session of Crypto'2000, available at `http://www.minrank.org/minrank/`.

[134] Nicolas Courtois : *Efficient Zero-knowledge authentication based on a linear algebra problem MinRank*. Asiacrypt 2001, December 9-13 2001, Gold Coast, Australia, LNCS Springer.

[135] Marc Girault : *A (non-practical) three pass identification protocol using coding theory ;* Advances in cryptology, AusCrypt'90, LNCS 453, pp. 265-272.

[136] Sami Harari. *A new authentication algorithm*. In Coding Theory and Applications, volume 388, pp.204-211, LNCS, 1989.

[137] Jacques Stern : *An alternative to the Fiat-Shamir protocol*; In Advances in Cryptology, Proceedings of Eurocrypt'89, LNCS 434, pp.173-180, Springer-Verlag.

[138] Jacques Stern : *A new identification scheme based on syndrome decoding*; In Advances in Cryptology, Proceedings of Crypto'93, LNCS 773, pp.13-21, Springer-Verlag.

[139] Pascal Véron, *Cryptanalysis of Harari's identification scheme*; Cryptography and Coding, vol. 1025, pp.264-269, LNCS 1025, Springer Verlag, 1995.

[140] Pascal Véron, *Problème SD, Opérateur Trace, Schémas d'Identification et Codes de Goppa*; PhD thesis in French, l'Université de Toulon et du Var, France, July 1995.

Autres schémas d'authentification Zéro-knowledge

La notion de Zéro-knowledge

[141] Mihir Bellare, Oded Goldreich : *On defining Proofs of Knowledge*; In Advances in Cryptology, Proceedings of Crypto'92, Springer-Verlag, LNCS vol. 740.

[142] Mihir Bellare, Oded Goldreich : *Proving Computational Ability*, draft, available at `http://www-cse.ucsd.edu/users/mihir/papers/poa.ps`

[143] Gilles Brassard, Claude Crépeau : *Sorting out zero-knowledge*, In Advances in Cryptology, Proceedings of Eurocrypt'89, pp.181, Springer-Verlag.

[144] Oded Goldreich, fragments of the book to come, `http://theory.lcs.mit.edu/~oded/frag.html`.

[145] Oded Goldreich, Silvio Micali, Adi Wigderson : *Proofs that yield nothing but their validity or All languages in NP have Zero knowledge proof systems.*; J. of the ACM, 1991, vol. 38, no. 1, pp.661-729.

[146] Oded Goldreich, Y. Oren. Definitions and properties of Zero-knowledge proof systems. Journal of Cryptology 1994, vol.7, no.1, pp.1-32.

[147] S. Goldwasser, S. Micali and C. Rackoff, *The knowledge Complexity of interactive proof systems*; SIAM Journal of computing, 1997, Vol. 6, No.1, pp.84.

[148] S. Hada, T. Tanaka : *On the Existence of 3-round Zero-knowledge protocols*. Crypto'98, extended and revised version is available at `http://philby.ucsd.edu/cryptolib/1999/99-09.html`

D'autres schémas Zéro-knowledge basés sur des problèmes NP-complets

[149] Jacques Patarin and P. Chauvaud : *Improved Algorithms for the Permuted Kernel Problem;* In Proceedings of CRYPTO'93, Springer-Verlag, Lecture Notes in Computer Science, LNCS 773, August 1993, pp. 391-402.

[150] David Pointcheval : *A new Identification Scheme Based on the Perceptrons Problem*; In Advances in Cryptology, Proceedings of Eurocrypt'95, LNCS 921, pp.319-328, Springer-Verlag.

[151] David Pointcheval : *Les preuves de connaissance et leurs preuves de sécurité*, PhD thesis, December 1996, Caen University, France.

[152] Guillaume Poupard : *A realistic Security Analysis of Identification Schemes Based on Combinatorial Problems ;* In Security in Communication Networks, SCN'96 (September 1996, Amalfi, Italy), European Transactions on Telecommunications ETT Vol. 8, No. 5, September/October 1997, pp. 471-480.

[153] Adi Shamir : *An efficient Identification Scheme Based on Permuted Kernels*, In Advances in Cryptology, Proceedings of Crypto'89, LNCS 435, pp.606-609, Springer-Verlag.

[154] Jacques Stern : *Designing identification schemes with keys of short size* ; In Advances in Cryptology, Proceedings of Crypto'94, LNCS 839, pp.164-73, Springer-Verlag.

[155] Le schéma Zéro-knowledge IP : voir [65]. Dans [88] on prouve que IP n'est malheureusement pas NP-dur.

Authentification Zéro-knowledge arithmétique

[156] M. J. Fisher, S. Micali, C. Rackoff : *A secure protocol for the oblivious transfer* ; Journal of Cryptology, 9(3), pp.191-195, 1996. Presented at Eurocrypt'84 but published twelve years later.

[157] T. Okamoto, E. Fujisaki :*Statistical Zero-knowledge Protocols to Prove Modular Polynomial Relations*, In IEICE Transactions, Japan, vol. E82-A No.1 pp.81-92, 1999/1

[158] Amos Fiat, Adi. Shamir : *How to prove yourself : Practical solutions to identification and signature problems*. In Advances in Cryptology, Crypto'86, pp. 186-194, Springer-Verlag, 1987.

[159] Louis Claude Guillou and Jean-Jacques Quisquater : *A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory* ; Advances in Cryptology, Eurocrypt '88, LNCS 330, Springer-Verlag, 1988, pp. 123-128.

[160] Jean-Jacques Quisquater and Louis C. Guillou : *The new Guillou-Quisquater Scheme ;*In Proceedings of the RSA 2000 conference, San Jose, USA. January 2000.

[161] Claus P. Schnorr : *Efficient identification and signatures for smart cards ;*In G. Brassard, editor, Advances in Cryptology, Crypto'89, LNCS 435, pp. 239-252, Springer-Verlag, 1990.